

THE COMPUTER-AIDED ANALYTIC PROCESS MODEL:  
APPENDIX TO THE OPERATIONS HANDBOOK FOR THE APM DEMONSTRATION PACKAGE

AD-A166 400

Ronald G. Shapiro  
Dunlap and Associates, East, Incorporated

for

ARI FIELD UNIT AT FORT BENNING, GEORGIA  
Joel D. Schendel, Acting Chief

TRAINING RESEARCH LABORATORY  
Seward Smith, Acting Director

DTIC FILE COPY

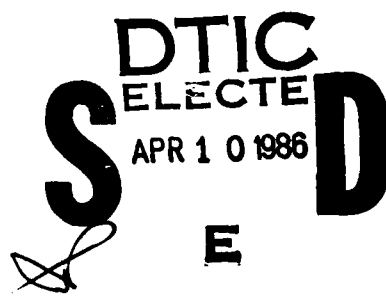


U. S. Army

Research Institute for the Behavioral and Social Sciences

January 1986

Approved for public release; distribution unlimited.



86 4 9 070

# U. S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the  
Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON  
Technical Director

WM. DARRYL HENDERSON  
COL, IN  
Commanding

---

Research accomplished under contract  
for the Department of the Army

Dunlap and Associates, East, Incorporated

This report, as submitted by the contractor, has been cleared for release to Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or other reference services such as the National Technical Information Service (NTIS). The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Research Note 86-07	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Computer-Aided Analytic Process Model: Appendix to the Operations Handbook for the APM Demonstration Package		5. TYPE OF REPORT & PERIOD COVERED Final Report May 1980 - February 1983
		6. PERFORMING ORG. REPORT NUMBER 293-26
7. AUTHOR(s) Ronald G. Shapiro		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0345
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dunlap and Associates East, Inc. 17 Washington Street Norwalk, CT 06584		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q263743A794
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Institute for the Behavioral and Social Sciences 5001 Eisenhower Ave., Alexandria, VA 22333-5600		12. REPORT DATE January 1986
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ARI Field Unit P.O. Box 2086 Fort Benning, Georgia 31905		13. NUMBER OF PAGES 318
		15. SECURITY CLASS. (of this report) Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES  Dr. Seward Smith, Contracting Officer's Representative		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer-Aided Model; Analytic Process Model; PASCAL; Computer Program; Apple		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Computer-Aided APM Demonstration Package provides the analyst with the opportunity to perform a thorough analysis of a system while the computer keeps track of the analysis and insures that the analyst examines the parts of the data base which are of interest. This is, however, a demonstration package which can only process small data bases. Because the package is implemented on an Apple II Plus, processing is relatively slow. An explanation of the APM, listings of the data sets derived using the APM and recommendations for further development of the APM appear in the companion volume--"The Analytic Process Model for System		

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. Design and Measurement: A Computer-Aided Tool for Analyzing Training Systems and Other Human-Machine Systems." A separate companion volume—"The Computer Aided Analytic Process Model: Operations Handbook for the APM Demonstration Package" is also available under separate cover. The present volume, which is an Appendix to the Operations Handbook, contains the actual PASCAL computer code listings. Disks containing this code and the data bases in machine-readable format are also available.

*Keywords: SEARCH computer program;  
PERFITEM computer program; PERFITEM  
MERCEATIP (Measurement Program); MERCEATIP (Measurement Program);  
MERCEATIP (Measurement Program); MERCEATIP (Measurement Program);  
MERCEATIP (Measurement Program); MERCEATIP (Measurement Program);  
MERCEATIP (Measurement Program); MERCEATIP (Measurement Program);*

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
<i>Ad</i>	



SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



## TABLE OF CONTENTS

	<u>Page</u>
I. EXECUTIVE SUMMARY	1
II. GREETING PROGRAM	3
III. PERFORMANCE ITEM PROGRAM (PERFITEM)	55
IV. MEASURES AND ATTRIBUTES PROGRAM (MEASATTR)	119
V. MEASUREMENT PURPOSE PROGRAM (MEASPURP)	179
VI. PRINT	223
VII. PACK	265
VIII. MISCELLANEOUS	307

## I. EXECUTIVE SUMMARY

This report is a supporting document to the Final Summary Report\* on an analytic process model (APM) for systems design and measurement. The present document, the Appendix to the Operations Manual for the computer-aided version of the APM,\*\* contains a listing of the Pascal code for the computer-aided model. The model was developed for the Army Research Institute (ARI) Field Unit, Fort Benning, over the period from March 1980 to February 1983.

The objective of the computer-aided APM is to provide a routinized, thorough, adaptive and efficient procedure to help testers, analysts and researchers develop design specifications and evaluation measures for any planned or existing human-machine system, and especially for any training system. The demonstration version of the computer-aided model, as described in this report, performs a sample of the routines expected in any ultimate version that may be developed in the future. Specifically, the demonstration model helps one to derive evaluation measures, but not design specifications. In addition, it contains data bases for training systems, but not for any other human-machine system. Finally, it contains data bases for only half of the six training subsystems (for design, enabling and delivery, but not for command, logistics or emplacement). For demonstration purposes, this development represents an appropriate and sufficient allocation of project resources, since the more significant effort was needed to develop the underlying concepts for both a feasible "manual" model and the computer-aided model. The demonstration model, using an Apple II Plus computer with two 5 $\frac{1}{4}$ -inch disk drives, programmed in PASCAL, can be exercised straight through, beginning with identifying the system and ending with a subset of its performance measures. Any larger capability than presently exists in the demonstration routine would require a computer with substantially greater capacity and speed.

Program listings are contained in Chapters 2-8. Chapter 2 contains a listing of the GREETING program which displays the title page, instructions and the analytic procedure menu. When the computer is turned on and the APM system disk is inserted, the title page is displayed first. Whenever the analyst decides to select a different analytic procedure, this program is loaded and the analytic procedure menu is displayed.

\*Bloom, R.F., Oates, J.F., Jr., Shapiro, R.G. and Hamilton, J.W. The Analytic Process Model for System Design and Measurement: A Computer-Aided Tool for Analyzing Training Systems and Other Human-Machine Systems. Norwalk, CT: Dunlap and Associates East, Inc., 28 February 1983. (Final Summary Report)

\*\*Shapiro, R.G., Bloom, R.F. and Oates, J.F., Jr. The Analytic Process Model For System Design and Measurement: Operations Handbook for the APM Demonstration Package. Norwalk, CT: Dunlap and Associates East, Inc., 28 February 1983.

Chapter 3 contains a listing of the PERFormance ITEM (PERFITEM) program. This program allows the analyst to add, reword, remove and print performance items (objectives, functional purposes and characteristics). Chapter 4 contains a listing of the MEASures and ATTRIBUTES (MEASATTR) program. MEASATTR allows the analyst to add, reword, remove and print attributes and measures for a given performance item. Chapter 5 contains a listing of the MEASurement PURPose (MEASPURP) program. MEASPURP allows the analyst to define a measurement purpose and associate (or disassociate) each of the characteristics with the measurement purpose.

Chapter 6 contains a listing of the PRINT program which allows the analyst to print the performance items, attributes and measures for a given measurement purpose, or an entire subsystem. Chapter 7 contains listings of the PACK program which arranges the data set for a given subsystem in order by item reference number, and packs the data sets so that any unused space is placed at the end of the data set so that it can be used.

Chapter 8 contains listings for a variety of programs which support the APM system. STARTUP asks the analyst to place the APM SYSTEM disk in Drive #1 at the appropriate time. GREETSHORT reminds the analyst to place the APM system disk in Drive #1 if he does not do so. BLOCKHELP and BLOCKINSTR set up the HELP, BRIEFHELP and INSTR data sets so that they are blocked efficiently for usage by the APM system. VIDPATCH modifies the SYSTEM.APPLE program for use with the VIDEX board. It only needs to be run once with each copy of the SYSTEM.APPLE program.

The operations handbook contains item-by-item directions for starting up and carrying out all the steps in the demonstration routine, schematic flow charts and miscellaneous information about the equipment and maintenance. Thus, it ought to be understood prior to reading the actual Pascal listings.

## **GREETING PROGRAM**

The greeting program presents the title page, instructions (if desired), establishes which system class, system and subsystem the analyst intends to use (while allowing for the possibility of creating new ones). The Greeting Program concludes by determining which analytic procedure is to be performed next. Whenever any analytic procedure is completed, the analytic procedure menu in this program is displayed to find out which program ought to be executed next.

```

1 1 1:D 1 ($L PRINTER:s)
2 1 1:D 1 ($S+s)
3 1 1:D 1 ($Program to greet user, instruct user, and set up table of systems and subsystems)
4 1 1:D 1 ($Ronald G. Shapiro V2.0 10/25/82s)
5 1 1:D 1
6 1 1:D 1 Program Greeting;
7 1 1:D 3
8 20 1:D 3
9 20 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
10 20 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
11 20 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
12 20 5:D 1 PROCEDURE SWAPON;
13 20 6:D 1 PROCEDURE SWAPOFF;
14 20 6:D 1
15 1 1:D 1 USES CHAINSTUFF;
16 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

17 1 1:D 3 (#P8)TYPE
18 1 1:D 3 PASSFILE=RECORD
19 1 1:D 3 CURSYS,CURSP,CURSUB,PAC:STRING(80);
20 1 1:D 3 NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
21 1 1:D 3 END;
22 1 1:D 3
23 1 1:D 3 SUBSYSFILE=RECORD
24 1 1:D 3 NSUBSYS: INTEGER;
25 1 1:D 3 SUBSYS:STRING(80);
26 1 1:D 3 END;
27 1 1:D 3
28 1 1:D 3 SPSYSFILE=RECORD
29 1 1:D 3 NSPSYS: INTEGER;
30 1 1:D 3 SPSYS:STRING(80);
31 1 1:D 3 END;
32 1 1:D 3
33 1 1:D 3 SYSFILE=RECORD
34 1 1:D 3 NSYSTEM: INTEGER;
35 1 1:D 3 SYSTEM:STRING(80);
36 1 1:D 3 END;
37 1 1:D 3
38 1 1:D 3 INSTRFILE=RECORD
39 1 1:D 3 LINE:ARRAY[1..20] OF STRING(80);
40 1 1:D 3 END;
41 1 1:D 3
42 1 1:D 3 HELPFILE=RECORD
43 1 1:D 3 LINE:ARRAY[1..10] OF STRING(80);
44 1 1:D 3 END;
45 1 1:D 3
46 1 1:D 3 FASTFILE=RECORD
47 1 1:D 3 PRINTIT:ARRAY[1..300]OF BOOLEAN;
48 1 1:D 3 END;
49 1 1:D 3

```

PASSFILE passes information about: 1) system class [CURSYS,NCURSYS] 2) system [CURSP,NCURSP] 3) subsystem [CURSB,NCURSB] 4) aspect [PAC,NPAC] from one program to another. Flag 1 is used to tell the GREETING program whether to begin with title page or analytic procedure list. Flags 2 and 3 are unused. SUBSYSFILE contains a list of the defined subsystems for each system. SPFILE contains a list of the defined systems for each system class. SYSFILE contains a list of the defined system classes. INSTRFILE contains the instructions. HELPFILE contains the help commands. FASTFILE allows fast printing of a measurement purpose if the measurement purpose had been printed before.

```

50 1 1:D 3 (86P8)VAR
51 1 1:D 3 XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB,LINE,REGLINE,ANSWER:STRING(80);
52 1 1:D 372 ANSHOLD,ANS2,ANS:CHAR;
53 1 1:D 375 DONE,OK,OVER,NEG:BOOLEAN;
54 1 1:D 379 MLENGTH,LLENGTH,PGE,I,NDATA,II,II2,J,K,L,M,N,NFUNPUR,NOBJECTIVE,
55 1 1:D 379 NPAC,NCURSYS,NCURSP,NCURSUB:INTEGER;
56 1 1:D 397 JHELP,HELP:INTEGER;
57 1 1:D 399 CORELAST,EII:INTEGER(8);
58 1 1:D 405 APMDSK:STRING(8);
59 1 1:D 410 NAMEFILETEST,NAMEFASTISSUE,FILESPPNAME,FRAME:STRING(24);
60 1 1:D 462
61 1 1:D 462 ASPECT:ARRAY[1..5] OF STRING(14);
62 1 1:D 502 SUBSYS,SPSYS,SYSTEM:ARRAY[1..10] OF STRING(80);
63 1 1:D 1732 SCRATCH:ARRAY[1..20] OF STRING(80);
64 1 1:D 2552 NSCRATCH:ARRAY[1..20] OF INTEGER;
65 1 1:D 2572 NSUBSYS,NSPSYS,NSYSTEM:ARRAY[1..10] OF INTEGER;
66 1 1:D 2602
67 1 1:D 2602 SYSLIST:FILE OF SYSFILE;
68 1 1:D 2944 SUBSYSLIST:FILE OF SUBSYSFILE;
69 1 1:D 3286 SPSYSLIST:FILE OF SPSYSFILE;
70 1 1:D 3628 PASSNODE:FILE OF PASSFILE;
71 1 1:D 4099 INSTFILE:FILE OF INSTRFILE;
72 1 1:D 5219 HELPER:FILE OF HELPFILE;
73 1 1:D 5929 PRNT:TEXT;
74 1 1:D 6230 FILETEST:TEXT;
75 1 1:D 6531 FASTISSUE:FILE OF FASTFILE;
76 1 1:D 7131

```

These strings, arrays and variables are used by the GREETING program.

```

77 1 2:D 1 (**P*)PROCEDURE KEY;FORWARD;
78 1 3:D 1 PROCEDURE KEYN;FORWARD;
79 1 4:D 1 PROCEDURE BRANCHOUT;FORWARD;
80 1 5:D 1 PROCEDURE SYSTEMFILES;FORWARD;
81 1 6:D 1 PROCEDURE S1;FORWARD;
82 1 7:D 1 PROCEDURE S2;FORWARD;
83 1 8:D 1 PROCEDURE S5;FORWARD;
84 1 9:D 1 PROCEDURE MENU;FORWARD;
85 1 10:D 1 PROCEDURE PROPERMAINDISK;FORWARD;
86 1 11:D 1 PROCEDURE OPENSFILES;FORWARD;
87 1 12:D 1 PROCEDURE GOSPSYSCREATE;FORWARD;
88 1 13:D 1 PROCEDURE SPYSYSCREATE;FORWARD;
89 1 14:D 1 PROCEDURE SPSYSTEMFILES;FORWARD;
90 1 15:D 1 PROCEDURE SUBSYSTEMFILES;FORWARD;
91 1 16:D 1 PROCEDURE PREPSPCREATE;FORWARD;
92 1 17:D 1 PROCEDURE HELPROUTINE;FORWARD;
93 1 18:D 1 PROCEDURE GOSUBCREATE;FORWARD;
94 1 18:D 1
95 1 18:D 1

```

These procedures are presented later on in the GREETING program.



```

96 1 19:0 1 (##P*)PROCEDURE KEYNPREP(HLP:INTEGER;MSG:STRING);
97 1 19:0 0 BEGIN
98 1 19:1 0 HELP:=HLP;
99 1 19:1 9 WRITE(MSG);
100 1 19:1 20 KEYN;
101 1 19:0 22 END;
102 1 19:0 34

```

KEYNPREP displays a one line message, then calls KEYN to read a number from the keyboard.

```

103 1 20:0 1 (800)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
104 1 20:0 0 BEGIN
105 1 20:1 0 HLP:=HLP;
106 1 20:1 9 REPEAT
107 1 20:2 9 WRITE(MSG);
108 1 20:2 20 KEY;
109 1 20:1 22 UNTIL (ANS='Y') OR (ANS='N');
110 1 20:0 35 END;
111 1 20:0 50

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

112 1 2:0 1 ($P$)PROCEDURE KEY;
113 1 2:0 0 BEGIN
114 1 2:0 0 ($R-$)
115 1 2:1 0 ANSWER:= '
116 1 2:1 24 REPEAT
117 1 2:2 24 READLN(ANSWER);
118 1 2:2 43 ANS:=ANSWER[1];
119 1 2:2 50 IF (ANS<>'Y')AND(ANS<>'N')AND(ANS<>'H')AND(ANS<>'y')AND
120 1 2:2 73 (ANS<>'n')AND(ANS<>'h') THEN
121 1 2:3 87 WRITELN('PLEASE RESPOND YES OR NO!');
122 1 2:2 132 IF ORD(ANS)>90 THEN
123 1 2:3 139 BEGIN
124 1 2:4 139 II2:=ORD(ANS)-32;
125 1 2:4 147 ANS:=CHR(II2);
126 1 2:3 153 END;
127 1 2:1 153 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H');
128 1 2:1 172 ($R+$)
129 1 2:1 172 IF ANS='H' THEN
130 1 2:2 179 HELPROUTINE;
131 1 2:0 181 END;
132 1 2:0 196

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key--it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

133 1 21:0 1 ($P$)PROCEDURE ANYKEY;
134 1 21:0 0 BEGIN
135 1 21:1 0 WRITELN(' ');
136 1 21:1 10 WRITELN('*** Please press any key to continue ***');
137 1 21:1 78 ($R-8)
138 1 21:1 78 READ(ANS);
139 1 21:1 89 ($R+8)
140 1 21:0 89 END;
141 1 21:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

142 1 3:0 1 (##P#)PROCEDURE KEYN;
143 1 3:0 1 VAR
144 1 3:0 1 ANSWER: STRING[40];
145 1 3:0 22 II: ARRAY[1..4] OF INTEGER;
146 1 3:0 26 OK: BOOLEAN;
147 1 3:0 27 IIO: INTEGER;
148 1 3:0 28
149 1 3:0 0 BEGIN
150 1 3:0 0 (##R-#)
151 1 3:1 0 OK:=TRUE;
152 1 3:1 3 REPEAT
153 1 3:2 3 REPEAT
154 1 3:3 3 I:=1;
155 1 3:3 8 ANSWER:= ' ;
156 1 3:3 35 READLN(ANSWER);
157 1 3:3 54 IF LENGTH(ANSWER)=0 THEN
158 1 3:4 62 WRITELN('Please enter the integer again');
159 1 3:2 112 UNTIL LENGTH(ANSWER)<>0;
160 1 3:2 120 IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
161 1 3:3 135 BEGIN
162 1 3:4 135 HELPROUTINE;
163 1 3:4 137 I:=999;
164 1 3:4 143 EXIT(KEYN);
165 1 3:3 147 END;
166 1 3:2 147 FOR I:=1 TO 4 DO
167 1 3:3 162 BEGIN
168 1 3:4 162 II[I]:=ORD(ANSWER[I])-48;
169 1 3:4 180 IF (II[I]<0) OR (II[I]>9) THEN
170 1 3:5 207 BEGIN
171 1 3:6 207 IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
172 1 3:7 229 BEGIN
173 1 3:8 229 OK:=FALSE;
174 1 3:8 232 WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
175 1 3:7 290 END;
176 1 3:5 290 END;
177 1 3:3 290 END;
178 1 3:1 300 UNTIL TRUE;
179 1 3:1 303 IIO:=II[I];
180 1 3:1 313 FOR I:=2 TO 4 DO
181 1 3:2 328 BEGIN
182 1 3:3 328 IF (II[I]=0) AND (II[I]<=9) THEN
183 1 3:4 355 IIO:=IIO*10+II[I];
184 1 3:2 372 END;
185 1 3:2 382 (##R+##)
186 1 3:1 382 I:=IIO;
187 1 3:0 387 END;
188 1 3:0 410

```

KEYN reads a 1 or 2 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 2 characters are typed, only 2 characters are read. The rest are ignored. If the character(s) are not positive integers, KEYN will display an appropriate warning and wait for a response.

```

189 1 22:0 1 ($P$)PROCEDURE SHOWALINE;
190 1 22:0 0 BEGIN
191 1 22:1 0 NLENGTH:=LENGTH(LINE);
192 1 22:1 7 WHILE LINE[NLENGTH]=' ' DO
193 1 22:2 18 NLENGTH:=NLENGTH-1;
194 1 22:1 28 IF NLENGTH<=LENGTH THEN
195 1 22:2 37 BEGIN
196 1 22:3 37 WRITE(LINE);
197 1 22:3 48 EXIT(SHOWALINE);
198 1 22:2 52 END;
199 1 22:1 52 L:=LENGTH;
200 1 22:1 58 WHILE LINE[L]<>' ' DO
201 1 22:2 69 L:=L-1;
202 1 22:1 79 L:=L-1;
203 1 22:1 87 REGLINE:=COPY(LINE,1,L);
204 1 22:1 104 L:=L+2;
205 1 22:1 112 WRITELN(REGLINE);
206 1 22:1 131 NLENGTH:=NLENGTH-L+1;
207 1 22:1 143 REGLINE:=COPY(LINE,L,NLENGTH);
208 1 22:1 162 WRITE(' ',REGLINE);
209 1 22:0 187 END;
210 1 22:0 204
210 1 22:0 204 ($I 05:HELPTXT.TEXT$)

```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```

211 1 23:0 1 (**P*)PROCEDURE PRNTHelp;
212 1 23:0 0 BEGIN
213 1 23:1 0 DONE:=FALSE;
214 1 23:1 4 REWRITE(PRNT,'PRINTER:');
215 1 23:1 25 PAGE(PRNT);
216 1 23:1 35 WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
217 1 23:1 97 WRITELN(PRNT,CHR(14),'Help File',chr(13));
218 1 23:1 146 PGE:=2;
219 1 23:1 150 REPEAT
220 1 23:2 150 SEEK(HELPER,PGE);
221 1 23:2 161 GET(HELPER);
222 1 23:2 169 PAGE(PRNT);
223 1 23:2 179 K:=PGE-1;
224 1 23:2 187 WRITELN(PRNT,'
225 1 23:2 239 Page ',K);
226 1 23:2 308 FOR J:=1 TO 10 DO
227 1 23:3 322 WRITELN(PRNT,HELPER^.LINE[J]);
228 1 23:2 362 IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
229 1 23:3 399 DONE:=TRUE;
230 1 23:2 403 PGE:=PGE+1;
231 1 23:1 411 UNTIL(DONE);
232 1 23:1 416 PAGE(PRNT);
233 1 23:1 426 CLOSE(PRNT);
234 1 23:0 435 END;
235 1 23:0 452

```

PRNTHelp prints the HELP file on the printer. It is called by HELPROUTINE.

```

236 1 17:0 1 ($P$)PROCEDURE HELPROUTINE;
237 1 17:0 0 BEGIN
238 1 17:0 0 ($I-$)
239 1 17:1 0 RESET(HELPER,'05:HELP');
240 1 17:1 18 ($I+$)
241 1 17:1 18 I:=IORESULT;
242 1 17:1 23 IF (I<>0) THEN
243 1 17:2 30 BEGIN
244 1 17:3 30 PAGE(OUTPUT);
245 1 17:3 40 WRITELN('UNFORTUNATELY, THE HELP FILE IS NOT AVAILABLE ON YOUR DISK');
246 1 17:3 118 WRITELN(' ');
247 1 17:3 136 WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
248 1 17:3 199 READ(ANS);
249 1 17:3 210 EXIT(HELPROUTINE);
250 1 17:2 214 END;
251 1 17:1 214 I:=0;
252 1 17:1 218 PGE:=HELP+1;
253 1 17:1 226 DONE:=FALSE;
254 1 17:1 230 REPEAT
255 1 17:2 230 SEEK(HELPER,PGE);
256 1 17:2 241 GET(HELPER);
257 1 17:2 249 PAGE(OUTPUT);
258 1 17:2 259 GOTOXY(73,0);
259 1 17:2 264 K:=PGE-1;
260 1 17:2 272 WRITELN('Page ',K);
261 1 17:2 309 GOTOXY(0,0);
262 1 17:2 314 FOR J:=1 TO 10 DO
263 1 17:3 328 WRITELN(HELPER^.LINE[J]);
264 1 17:2 368 IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
265 1 17:3 405 DONE:=TRUE;
266 1 17:2 409 WRITELN(' ');
267 1 17:2 427 WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
268 1 17:2 496 WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW ANOTHER
PAGE****');
269 1 17:2 589 WRITE ('****PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE HELP ROUTINE****');
270 1 17:2 664 PGE:=PGE+1;
271 1 17:2 672 ($R-$)
272 1 17:2 672 ANSWER:= ' ';
273 1 17:2 689 READLN(ANSWER);
274 1 17:2 708 page(output);

```

HELPROUTINE displays appropriate help commands when it is called by KEY or KEYN. HELPROUTINE knows which HELP to display because the calling program places the appropriate help page number into HELP. Once the analyst sees the first help message, he/she can ask for other help messages by typing in the page number of the desired help messages. Note that the HELP file is made by editing a series of files (HELP1 . . . HELPN) using the Apple editor. Then, they are processed by the BLOCKHELP program (see Chapter VIII). The HELP file produced by BLOCKHELP is suitable for use with the HELPROUTINE. HELPROUTINE "knows" it has hit the last page of the file because the word "conclusion" appears on the second line of the last page.



```

275 1 17:2 718      IF ORD(ANSWER[1])=27 THEN
276 1 17:3 726      BEGIN
277 1 17:4 726          CLOSE(HELPER);
278 1 17:4 735          ($R+$)
279 1 17:4 735          EXIT (HELPROUTINE);
280 1 17:4 739          ($R-$)
281 1 17:3 739          END;
282 1 17:2 739      IF (ANSWER[1]='0') AND (ANSWER[1]<='9') THEN
283 1 17:3 754          BEGIN
284 1 17:4 754              PGE:=ORD(ANSWER[1])-48;
285 1 17:4 763              IF (ANSWER[2]='0') AND (ANSWER[2]<='9') THEN
286 1 17:5 778                  PGE:=PGE*10 + ORD(ANSWER[2])-48;
287 1 17:4 793              PGE:=PGE+1;
288 1 17:4 801              IF PGE<2 THEN
289 1 17:5 808                  PGE:=2;
290 1 17:4 812              DONE:=FALSE;
291 1 17:3 816              END;
292 1 17:1 816          UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
293 1 17:1 835      PAGE(OUTPUT);
294 1 17:1 845      PREPKEY(2,'Would you like to print the help file?');
295 1 17:1 889      IF ANS='Y' THEN
296 1 17:2 896          BEGIN
297 1 17:3 896              KEYNPREP(2,'How many copies? ');
298 1 17:3 919              FOR N:=1 TO 1 DO
299 1 17:4 935                  PRNTHLP;
300 1 17:3 947              WRITELN('DONE');
301 1 17:2 971              END;
302 1 17:1 971          CLOSE(HELPER)
303 1 17:1 980          ($R+$)
304 1 17:0 980          END;
305 1 17:0 1000
306 1 17:0 1000
307 1 17:0 1000 ($I 05:HELPTXT.TEXT$)
308 1 17:0 1000

```

See previous page for program description.

```

309 1 24:D 1 ($$P$)PROCEDURE QUIT;
310 1 24:0 0 BEGIN
311 1 24:1 0 PAGE(OUTPUT);
312 1 24:1 10 REPEAT
313 1 24:2 10 write('Would you like to return to title page?');
314 1 24:2 61 help:=2;
315 1 24:2 65 key
316 1 24:1 65 until (ans='Y') or (ans='N');
317 1 24:1 80 if ans='Y' then
318 1 24:2 87 begin
319 1 24:3 87 setchain('greeting');
320 1 24:3 101 passnode^.flag1:=0;
321 1 24:3 109 branchout;
322 1 24:3 111 exit(program);
323 1 24:2 115 end;
324 1 24:1 115 REPEAT
325 1 24:2 115 write('Would you like to turn off computer for now?');
326 1 24:2 171 help:=2;
327 1 24:2 175 key
328 1 24:1 175 until (ans='Y') or (ans='N');
329 1 24:1 190 if ans='Y' then
330 1 24:2 197 begin
331 1 24:3 197 passnode^.flag1:=0;
332 1 24:3 205 branchout;
333 1 24:3 207 page(output);
334 1 24:3 217 writeln('bye...');
335 1 24:3 243 writeln('');
336 1 24:3 263 writeln(' I hope to see you again very soon!');
337 1 24:3 318 writeln(' ');
338 1 24:3 336 writeln('You say now!');
339 1 24:3 368 writeln(' 1. Remove the disks. ');
340 1 24:3 409 writeln(' 2. Turn off printer. ');
341 1 24:3 450 writeln(' 3. Turn off computer. ');
342 1 24:3 492 writeln(' 4. Turn off this display screen. ');
343 1 24:3 545 OVER:=FALSE;
344 1 24:3 549 REPEAT
345 1 24:4 549 I:=1
346 1 24:3 549 UNTIL OVER=TRUE;
347 1 24:2 561 end;
348 1 24:1 561 REPEAT

```

QUIT asks the analyst what he/she wants to do next 1) return to the title page, 2) turn off the computer, 3) access the Apple operating system. If options 2 or 3 are selected, it says bye . . . and displays some helpful advice. If option 1 is selected, the computer then goes into an infinite loop, whereas if option 3 is selected, the analyst gains control of the Apple operating system. If the analyst selects no option, then he/she is sent back into the APM demonstration package, approximately where he/she left off.

```

349 1 24:2 561      write('Would you like to leave model and access computer operating system?');
350 1 24:2 640      help:=2;
351 1 24:2 644      key
352 1 24:1 644      until (ans='Y') or (ans='N');
353 1 24:1 659      if ans='Y' then
354 1 24:2 666          begin
355 1 24:3 666              passnode^.flag1:=0;
356 1 24:3 674              branchout;
357 1 24:3 676              PAGE(OUTPUT);
358 1 24:3 686              writeln('bye...');
359 1 24:3 712              writeln('');
360 1 24:3 732              writeln(' I hope to see you again very soon!');
361 1 24:3 788              writeln('');
362 1 24:3 808              writeln(' You are now on your own with the Apple OS');
363 1 24:3 870              writeln('');
364 1 24:3 890              writeln(' good luck...');
365 1 24:3 923              for i:=1 to 1000 do
366 1 24:4 939                  i:=i+10;
367 1 24:3 957              exit(program);
368 1 24:2 961          end;
369 1 24:1 961      writeln('Since you have selected not to exit from this program, I will ');
370 1 24:1 1043      writeln(' send you back to where you left off as soon as you press any key. ');
371 1 24:1 1129      ($$r-8)
372 1 24:1 1129      read(ans);
373 1 24:1 1140      ($$r+8)
374 1 24:0 1140      END;
375 1 24:0 1166

```

See previous page for program description.

```

376 1 25:0 1 (86P)PROCEDURE HELLO;
377 1 25:0 0 BEGIN
378 1 25:1 0 PAGE(OUTPUT);
379 1 25:1 10 WRITELN(' ');
380 1 25:1 28 WRITELN(' ');
381 1 25:1 46 WRITELN(' ');
382 1 25:1 64 WRITELN(' ');
383 1 25:1 82 WRITELN(' ');
384 1 25:1 156 WRITELN(' ');
385 1 25:1 174 WRITELN(' ');
386 1 25:1 249 WRITELN(' ');
387 1 25:1 267 WRITELN(' ');
388 1 25:1 343 WRITELN(' ');
389 1 25:1 361 WRITELN(' ');
390 1 25:1 379 WRITELN(' ');
391 1 25:1 397 WRITELN(' ');
392 1 25:1 415 WRITELN(' ');
393 1 25:1 433 WRITELN(' ');
394 1 25:1 509 WRITELN(' ');
395 1 25:1 599 WRITELN(' ');
396 1 25:1 645 WRITELN(' ');
397 1 25:1 663 WRITELN(' ');
398 1 25:1 681 WRITE(' ');
399 1 25:1 746 (86R-8)
400 1 25:1 746 READ(ANS);
401 1 25:1 757 (86R+8)
402 1 25:0 757 END;
403 1 25:0 770
404 1 25:0 770

```

AN ANALYTIC PROCESS MODEL FOR';  
 SYSTEMS DESIGN AND MEASUREMENT:');  
 APPLICATIONS TO TRAINING SYSTEMS');

Prepared for: ARI Field Unit, Fort Benning, Georgia');  
 Prepared by: Dunlap & Associates East, Inc., Norwalk, Conn');  
 Date: 25 October 1982');

PLEASE PRESS ANY KEY TO BEGIN');

HELLO displays the title page.

```

405 1 26:D 1 (%%P*)PROCEDURE PRINTINSTRUCTIONS;
406 1 26:0 0 BEGIN
407 1 26:1 0 DONE:=FALSE;
408 1 26:1 4 REWRITE(PRNT,'PRINTER:');
409 1 26:1 25 PAGE(PRNT);
410 1 26:1 35 WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
411 1 26:1 97 WRITELN(PRNT,CHR(14),'Instructions',chr(13));
412 1 26:1 149 PGE:=2;
413 1 26:1 153 REPEAT
414 1 26:2 153 SEEN(INSTFILE,PGE);
415 1 26:2 164 GET(INSTFILE);
416 1 26:2 172 PAGE(PRNT);
417 1 26:2 182 K:=PGE-1;
418 1 26:2 190 WRITELN(PRNT,'
419 1 26:2 242 Page ',K);
420 1 26:2 311 FOR J:=1 TO 20 DO
421 1 26:3 325 WRITELN(PRNT,INSTFILE^.LINE[J]);
422 1 26:2 365 IF COPY(INSTFILE^.LINE[2],2,10)='conclusion' THEN
423 1 26:3 402 DONE:=TRUE;
424 1 26:2 406 PGE:=PGE+1;
425 1 26:1 414 UNTIL(DONE);
426 1 26:1 419 PAGE(PRNT);
427 1 26:1 429 CLOSE(PRNT);
428 1 26:0 438 END;
429 1 26:0 454

```

PRINTINSTRUCTIONS prints the instructions on the printer when it is called by INSTRUCTIONS. (It is nearly identical in structure to PRINTHELP.)

```

430 1 27:D 1 (##P*)PROCEDURE INSTRUCTIONS;
431 1 27:0 0 BEGIN
432 1 27:0 0 (##I-#)
433 1 27:1 0 RESET(INSTFILE,'APHUTL:INSTRUCT');
434 1 27:1 26 (##I+#)
435 1 27:1 26 I:=IORESULT;
436 1 27:1 31 IF I=9 THEN
437 1 27:2 38 BEGIN
438 1 27:3 38 PROPERMAINDISK;
439 1 27:3 40 INSTRUCTIONS;
440 1 27:3 42 EXIT(INSTRUCTIONS);
441 1 27:2 46 END;
442 1 27:1 46 IF (I<>0)AND(I<>9) THEN
443 1 27:2 59 BEGIN
444 1 27:3 59 PAGE(OUTPUT);
445 1 27:3 69 WRITELN('UNFORTUNATELY, INSTRUCTION FILE IS NOT AVAILABLE ON YOUR DISK');
446 1 27:3 154 WRITELN(' ');
447 1 27:3 172 WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
448 1 27:3 235 READ(ANS);
449 1 27:3 246 EXIT(INSTRUCTIONS);
450 1 27:2 250 END;
451 1 27:1 250 I:=0;
452 1 27:1 254 PGE:=2;
453 1 27:1 258 DONE:=FALSE;
454 1 27:1 262 REPEAT
455 1 27:2 262 SEEK(INSTFILE,PGE);
456 1 27:2 273 GET(INSTFILE);
457 1 27:2 281 PAGE(OUTPUT);
458 1 27:2 291 GOTOXY(73,0);
459 1 27:2 296 K:=PGE-1;
460 1 27:2 304 WRITELN('Page ',K);
461 1 27:2 341 GOTOXY(0,0);
462 1 27:2 346 FOR J:=1 TO 20 DO
463 1 27:3 360 WRITELN(INSTFILE^.LINE[J]);
464 1 27:2 400 IF COPY(INSTFILE^.LINE[2],2,10)='conclusion' THEN
465 1 27:3 437 DONE:=TRUE;
466 1 27:2 441 WRITELN(' ');
467 1 27:2 459 WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
468 1 27:2 528 WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW ANOTHER
PAGE****');

```

INSTRUCTIONS displays the instructions. Functionally, it is virtually identical to HELPROUTINE.

```

469 1 27:2 621 WRITE ('***PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE INSTRUCTIONS***');
470 1 27:2 696 PGE:=PGE+1;
471 1 27:2 704 ($R-$)
472 1 27:2 704 ANSWER:=' ';
473 1 27:2 721 READLN(ANSWER);
474 1 27:2 740 page(output);
475 1 27:2 750 IF ORD(ANSWER[1])=27 THEN
476 1 27:3 758 BEGIN
477 1 27:4 758 CLOSE(INSTFILE);
478 1 27:4 767 ($R+$)
479 1 27:4 767 EXIT (INSTRUCTIONS);
480 1 27:4 771 ($R-$)
481 1 27:3 771 END;
482 1 27:2 771 IF (ANSWER[1]>='0') AND (ANSWER[1]<='9') THEN
483 1 27:3 786 BEGIN
484 1 27:4 786 PGE:=ORD(ANSWER[1])-48;
485 1 27:4 795 IF (ANSWER[2]>='0') AND (ANSWER[2]<='9') THEN
486 1 27:5 810 PGE:=PGE*10 + ORD(ANSWER[2])-48;
487 1 27:4 825 PGE:=PGE+1;
488 1 27:4 833 IF PGE<2 THEN
489 1 27:5 840 PGE:=2;
490 1 27:4 844 DONE:=FALSE;
491 1 27:3 848 END;
492 1 27:1 848 UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
493 1 27:1 867 PAGE(OUTPUT);
494 1 27:1 877 PREPKEY(2,'Would you like to print these instructions?');
495 1 27:1 926 IF ANS='Y' THEN
496 1 27:2 933 BEGIN
497 1 27:3 933 KEYNPREP(2,'How many copies? ');
498 1 27:3 956 FOR N:=1 TO 1 DO
499 1 27:4 972 PRINTINSTRUCTIONS;
500 1 27:3 984 WRITELN('DONE');
501 1 27:2 1008 END;
502 1 27:1 1008 CLOSE(INSTFILE)
503 1 27:1 1017 ($R+$)
504 1 27:0 1017 END;
505 1 27:0 1038

```

See previous page for program description.

```

506 1 4:D 1 (##P*)PROCEDURE BRANCHOUT;
507 1 4:0 0 BEGIN
508 1 4:0 0 (##I-#)
509 1 4:1 0 RESET(PASSNODE,'PASSTHRU');
510 1 4:1 19 (##I+#)
511 1 4:1 19 IF(IORRESULT<>0) THEN
512 1 4:2 25 WRITE('SERIOUS ERROR -- NO FILE PASSTHRU AT BRANCHOUT')
513 1 4:1 83 ELSE
514 1 4:2 85 BEGIN
515 1 4:3 85 PASSNODE^.CURSYS:=CURSYS;
516 1 4:3 95 PASSNODE^.CURSUB:=CURSUB;
517 1 4:3 104 PASSNODE^.CURSP:=CURSP;
518 1 4:3 114 PASSNODE^.PAC:=PAC;
519 1 4:3 122 PASSNODE^.NCURSYS:=NCURSYS;
520 1 4:3 132 PASSNODE^.NCURSUB:=NCURSUB;
521 1 4:3 142 PASSNODE^.NCURSP:=NCURSP;
522 1 4:3 152 PASSNODE^.NPAC:=NPAC;
523 1 4:3 162 PASSNODE^.FLAG1:=0;
524 1 4:3 170 PUT(PASSNODE);
525 1 4:3 178 IF EOF(PASSNODE) THEN
526 1 4:4 188 WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
527 1 4:3 248 CLOSE(PASSNODE,LOCK);
528 1 4:2 257 END;
529 1 4:0 257 END;
530 1 4:0 272

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.



```

531 1 28:D 1 (##P*)PROCEDURE BRANCHIN;
532 1 28:0 0 BEGIN
533 1 28:0 0 (##I-8)
534 1 28:1 0 RESET(PASSNODE,'PASSTHRU');
535 1 28:1 19 (##I+8)
536 1 28:1 19 IF IORESULT<>0 THEN
537 1 28:2 25 BEGIN
538 1 28:3 25 REWRITE(PASSNODE,'PASSTHRU');
539 1 28:3 46 PASSNODE^.CURSYS:='';
540 1 28:3 56 PASSNODE^.CURSUB:='';
541 1 28:3 66 PASSNODE^.PAC:='';
542 1 28:3 74 PASSNODE^.NCURSYS:=0;
543 1 28:3 82 PASSNODE^.NCURSP:=0;
544 1 28:3 90 PASSNODE^.NCURSUB:=0;
545 1 28:3 98 PASSNODE^.NPAC:=0;
546 1 28:3 106 PUT(PASSNODE);
547 1 28:3 114 IF EOF(PASSNODE) THEN
548 1 28:4 124 WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
549 1 28:3 184 CLOSE(PASSNODE,LOCK);
550 1 28:3 193 RESET(PASSNODE,'PASSTHRU')
551 1 28:2 214 END;
552 1 28:1 214 GET(PASSNODE);
553 1 28:1 222 CURSYS:=PASSNODE^.CURSYS;
554 1 28:1 232 CURSP:=PASSNODE^.CURSP;
555 1 28:1 242 CURSUB:=PASSNODE^.CURSUB;
556 1 28:1 251 PAC:=PASSNODE^.PAC;
557 1 28:1 259 NCURSYS:=PASSNODE^.NCURSYS;
558 1 28:1 268 NCURSUB:=PASSNODE^.NCURSUB;
559 1 28:1 277 NPAC:=PASSNODE^.NPAC;
560 1 28:1 286 CLOSE(PASSNODE,LOCK);
561 1 28:0 295 END;
562 1 28:0 310

```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```

563 1 29:0 1  (##Ps)PROCEDURE MAKEDISK;
564 1 29:0 0  BEGIN
565 1 29:1 0    PREPKEY(54,'Would you like to prepare a new data disk for this subsystem?');
566 1 29:1 67   IF ANS='N' THEN
567 1 29:2 74     BEGIN
568 1 29:3 74       MENU;
569 1 29:3 76       EXIT(MAKEDISK);
570 1 29:2 80     END;
571 1 29:1 80     WRITELN('I can neither format nor name a disk, so I will tell you what to do',
572 1 29:1 159      chr(13),' then you will have the opportunity to do it.',chr(13),
573 1 29:1 237      ' 1. Turn to page 184 of the APPLE PASCAL Operating System Reference
574 1 29:1 335      Manual.',chr(13),
575 1 29:1 415      ' 2. Insert Apple 3 in drive 1, and follow the directions.',chr(13),
576 1 29:1 492      ' 3. When finished, turn to page 33 of the same manual.',chr(13),
577 1 29:1 543      ' 4. Enter the FILER program.',chr(13),
578 1 29:1 622      ' 5. Turn to page 45 and run the change program, changing',chr(13),
579 1 29:1 703      '     BLANK: to the new name followed by a colon. The new',chr(13),
580 1 29:1 787      '     name consists of the first 3 letters of the system name',chr(13),
581 1 29:1 870      '     followed by the first 3 letters of the subsystem name.',chr(13),
582 1 29:1 931      ' 6. Good Luck--I will see you again soon');
583 1 29:0 935     EXIT(PROGRAM);
584 1 29:0 948     END;

```

MAKEDISK tells analyst how to format a new disk if no disk is available for the subsystem requested.

```

585 1 10:0 1 (##P)PROCEDURE PROPERMAINDISK;
586 1 10:0 0 BEGIN
587 1 10:1 0 PAGE(OUTPUT);
588 1 10:1 10 REPEAT
589 1 10:1 10 (##I-8)
590 1 10:2 10 RESET(INSTFILE,'APMUTL:INSTRUCT');
591 1 10:2 36 (##I+8)
592 1 10:2 36 K:=IORESULT;
593 1 10:2 41 IF K=0 THEN
594 1 10:3 48 CLOSE(INSTFILE);
595 1 10:2 57 IF K=9 THEN
596 1 10:3 64 BEGIN
597 1 10:4 64 WRITELN('Please place the APM UTILITY disk (APMUTIL) in drive # 2.');

```

PROPERMAINDISK checks to be sure that the APMUTILITY disk is in Drive #2 when it is needed.

```

603 1 30:0 1 (##P)PROCEDURE PROPERDISK;
604 1 30:0 0 BEGIN
605 1 30:1 0 OK:=TRUE;
606 1 30:1 4 PAGE(OUTPUT);
607 1 30:1 14 WRITELN('System class: ',CURSYS);
608 1 30:1 40 WRITELN('System: ',CURSP);
609 1 30:1 100 WRITELN('Subsystem: ',CURSUB);
610 1 30:1 142 WRITELN(' ');
611 1 30:1 160 APHDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),'');
612 1 30:1 245 NAMEFILETEST:=CONCAT(APHDSK,'TEST');
613 1 30:1 277 REPEAT
614 1 30:1 277 (##I-8)
615 1 30:2 277 RESET(FILETEST,NAMEFILETEST);
616 1 30:2 288 (##I+8)
617 1 30:2 288 K:=IORESULT;
618 1 30:2 293 IF K=9 THEN
619 1 30:3 300 BEGIN
620 1 30:4 300 HELP:=2;
621 1 30:4 304 WRITELN('If the disk for this system class, system, and subsystem is'.
622 1 30:4 375 chr(13),' available, place it in Drive # 2 and type Y (retrn).',
623 1 30:4 459 chr(13),' Otherwise, type N (return).');
624 1 30:4 518 KEY;
625 1 30:4 520 IF ANS='Y' THEN
626 1 30:5 527 BEGIN
627 1 30:6 527 PROPERDISK;
628 1 30:6 529 EXIT(PROPERDISK);
629 1 30:5 533 END;
630 1 30:4 533 IF (ANS='N') OR (ANS='n') THEN
631 1 30:5 546 OK:=FALSE;
632 1 30:3 550 END;
633 1 30:1 550 UNTIL (K<>9) OR (OK=FALSE);
634 1 30:1 564 IF OK=FALSE THEN
635 1 30:2 572 MAKEDISK;
636 1 30:0 574 END;
637 1 30:0 590

```

PROPERDISK checks to be sure that the appropriate disk for the system class, system and subsystem selected is in Drive #2 before branching to another program.

```

638 1 31:D 1 (%%*)PROCEDURE REMOVEFASTISSUE;
639 1 31:0 0 BEGIN
640 1 31:0 0 (%%I-*)
641 1 31:1 0 NAMEFASTISSUE:=CONCAT(APMDSA,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'FA');
642 1 31:1 95 RESET(FASTISSUE,NAMEFASTISSUE);
643 1 31:1 106 CLOSE(FASTISSUE,PURGE);
644 1 31:1 113 (%%I+*)
645 1 31:0 113 END;
646 1 31:0 126

```

REMOVEFASTISSUE deletes the FASTISSUE file whenever PACKDATA is run.  
 [A new FASTISSUE file will be created the next time the PRINT program is run.]

```

647 1 32:D 1 (ssp)PROCEDURE SHOWMENU;
648 1 32:0 0 BEGIN
649 1 32:1 0 page(output);
650 1 32:1 10 WRITELN('System class: ',cursys);
651 1 32:1 56 WRITELN('System: ',cursp);
652 1 32:1 96 WRITELN('Subsystem: ',cursub);
653 1 32:1 138 WRITELN(' ');
654 1 32:1 156 WRITELN('You may perform the following analytic procedures:');
655 1 32:1 226 writeln(' ');
656 1 32:1 244 writeln( '1. Add, modify, or delete performance items',chr(13),
657 1 32:1 310 '2. Add, modify, or delete measurable attributes or measures',chr(13),
658 1 32:1 392 '3. Add, modify, or delete measurement purposes',chr(13),
659 1 32:1 461 '4. Print out selected results from your analysis',chr(13),
660 1 32:1 532 '5. Pack your disk files most efficiently (a slow process)',chr(13),
661 1 32:1 612 '6. Change System class, System, and/or Subsystem to be analyzed',chr(13),
662 1 32:1 698 '7. Review Instructions',chr(13),
663 1 32:1 743 '8. Stop for now',chr(13),chr(13));
664 1 32:0 799 END;
665 1 32:0 812

```

SHOWMENU displays the list of analytic procedures available.

```

666 1 9:0 1 (80P8)PROCEDURE MENU;
667 1 9:0 0 BEGIN
668 1 9:1 0 SHOWMENU;
669 1 9:1 2 REPEAT
670 1 9:2 2 KEYNPREP(5,'Which would you like to do?');
671 1 9:2 35 IF I=999 THEN
672 1 9:3 44 BEGIN
673 1 9:4 44 MENU;
674 1 9:4 46 EXIT(MENU);
675 1 9:3 50 END;
676 1 9:2 50 IF (I>8) OR (I<1) THEN
677 1 9:3 63 Writeln('Please type an integer between 1 and 8');
678 1 9:1 121 UNTIL (I>0) AND (I<9);
679 1 9:1 134 IF I<6 THEN
680 1 9:2 141 BEGIN
681 1 9:3 141 PROPERDISK;
682 1 9:3 143 IF K=9 THEN
683 1 9:4 150 BEGIN
684 1 9:5 150 Writeln('Options 1 to 5 are not available, because you are not able',
685 1 9:5 220 chr(13), 'to insert the appropriate disk. Please select',
686 1 9:5 288 chr(13), 'Option 6, 7, or 8 when the menu reappears');
687 1 9:5 359 ANYKEY;
688 1 9:5 361 MENU;
689 1 9:4 363 END;
690 1 9:2 363 END;
691 1 9:2 363
692 1 9:1 363 CASE 1 OF
693 1 9:1 368 1: BEGIN
694 1 9:3 368 SETCHAIN('PERFITM');
695 1 9:3 382 EXIT(PROGRAM);
696 1 9:2 386 END;
697 1 9:1 388 2: BEGIN
698 1 9:3 388 SETCHAIN('MEASATTR');
699 1 9:3 402 EXIT(PROGRAM);
700 1 9:2 406 END;
701 1 9:1 408 3: BEGIN
702 1 9:3 408 SETCHAIN('MEASPURP');
703 1 9:3 422 EXIT(PROGRAM);
704 1 9:2 426 END;
705 1 9:1 428 4: BEGIN

```

MENU calls SHOWMENU to display the list of analytic procedures available. Then, menu calls KEYN to find out which analytic procedure the analyst wishes to perform. It then sets up the Apple chaining program to execute the desired procedure. Then, it branches to that procedure.

```

706 1 9:3 428      SETCHAIN('PRINT');
707 1 9:3 439      EXIT(PROGRAM);
708 1 9:2 443      END;
709 1 9:1 445      5: BEGIN
710 1 9:3 445        REMOVEFASTISSUE;
711 1 9:3 447        SETCHAIN('PACDATA');
712 1 9:3 461        EXIT(PROGRAM);
713 1 9:2 465        END;
714 1 9:1 467      6: BEGIN
715 1 9:3 467        PROPERMAINDISK;
716 1 9:3 469        SYSTEMFILES;
717 1 9:3 471        SPSYSTEMFILES;
718 1 9:3 473        SUBSYSTEMFILES;
719 1 9:3 475        BRANCHOUT;
720 1 9:3 477        MENU;
721 1 9:3 479        EXIT(MENU);
722 1 9:2 483        END;
723 1 9:1 485      7: BEGIN
724 1 9:3 485        PROPERMAINDISK;
725 1 9:3 487        INSTRUCTIONS;
726 1 9:3 489        MENU;
727 1 9:2 491        END;
728 1 9:1 493      8: BEGIN
729 1 9:3 493        QUIT;
730 1 9:3 495        MENU;
731 1 9:3 497        EXIT(MENU);
732 1 9:2 501        END;
733 1 9:1 503      END;
734 1 9:0 526      END;
735 1 9:0 548

```

See previous page for program description.



```

736 1 33:0 1 (**P*)PROCEDURE SUBSYSCREATE;
737 1 33:0 0 BEGIN
738 1 33:1 0 REPEAT
739 1 33:2 0 I:=0;
740 1 33:2 4 REPEAT
741 1 33:3 4 I:=I+1;
742 1 33:2 12 UNTIL(I=10) OR (SUBSYS[I]='');
743 1 33:2 38 IF I=10 THEN
744 1 33:3 45 BEGIN
745 1 33:4 45 WRITELN('***WARNING SYST: NO ROOM FOR MORE SUBSYSTEMS FOR SYSTEM CLASS',CURSYS);
746 1 33:4 138 ANYKEY;
747 1 33:4 140 EXIT(SUBSYSCREATE)
748 1 33:3 144 END
749 1 33:2 144 ELSE
750 1 33:3 146 GOSUBCREATE;
751 1 33:1 148 UNTIL OK;
752 1 33:0 153 END;
753 1 33:0 170

```

SUBSYSCREATE enters subsystem names into the SUBSYSFILE for a given system class and system.

```

754 1 18:0 1(##P)PROCEDURE GOSUBCREATE;
755 1 18:0 0 BEGIN
756 1 18:1 0 WRITE('What is the name of your subsystem?');
757 1 18:1 47 SUBSYS[I]:='';
758 1 18:1 65 REPEAT
759 1 18:2 65 READLN(SUBSYS[I]);
760 1 18:2 95 IF SUBSYS[I]='' THEN
761 1 18:3 115 EXIT(SUBSYSCREATE);
762 1 18:2 119 IF LENGTH(SUBSYS[I])<5 THEN
763 1 18:3 138 WRITE('Subsystem name must contain at least 5 letters--',CHR(13),
764 1 18:3 208 'Please type a new name:');
765 1 18:2 243 K:=POS(' ',SUBSYS[I]);
766 1 18:2 268 IF (K>0) AND (K<6) THEN
767 1 18:3 281 WRITE('None of the first five characters of subsystem name can be blank--',chr(13),
768 1 18:3 349 'Please type a new name:');
769 1 18:1 404 UNTIL (LENGTH(SUBSYS[I])>=5) AND ((K<1) OR (K>5));
770 1 18:1 435 NSUBSYS[I]:=I;
771 1 18:1 452 WRITELN('Subsystem ',SUBSYS[I], ' is member number ',NSUBSYS[I],
772 1 18:1 549 CHR(13), ' of system ',CURSP);
773 1 18:1 602 RESET(SUBSYSLIST,FRAME);
774 1 18:1 615 SEEK(SUBSYSLIST,I);
775 1 18:1 626 SUBSYSLIST^.NSUBSYS:=NSUBSYS[I];
776 1 18:1 644 SUBSYSLIST^.SUBSYS:=SUBSYS[I];
777 1 18:1 664 PUT (SUBSYSLIST);
778 1 18:1 672 CLOSE(SUBSYSLIST,LOCK);
779 1 18:1 681 WRITELN(' ');
780 1 18:1 699 REPEAT
781 1 18:2 699 WRITELN('Would you like to proceed with the analysis of system class ',CURSYS,
782 1 18:2 783 ' ',CHR(13),'system ',CURSP, ' ;subsystem ',SUBSYS[I],'?');
783 1 18:2 898 HELP:=33;
784 1 18:2 902 KEY;
785 1 18:1 904 UNTIL (ANS='Y') OR (ANS='N');
786 1 18:1 917 IF ANS='Y' THEN
787 1 18:2 924 BEGIN
788 1 18:3 924 CURSUB:=SUBSYS[I];
789 1 18:3 941 NCURSUB:=NSUBSYS[I];
790 1 18:3 958 PASSNODE^.FLAG1:=0;
791 1 18:3 966 BRANCHOUT;
792 1 18:3 968 MENU;
793 1 18:2 970 END;

```

GOSUBCREATE is a continuation of SUBSYSCREATE.

```

794 1 18:1 970 REPEAT
795 1 18:2 970 WRITE('Would you like to add more subsystems to system ',CURSP,'?');
796 1 18:2 1052 HELP:=S4;
797 1 18:2 1056 KEY
798 1 18:1 1056 UNTIL (ANS='Y') OR (ANS='N');
799 1 18:1 1071 OK:=TRUE;
800 1 18:1 1075 IF ANS='Y' THEN
801 1 18:2 1082 OK:=FALSE
802 1 18:1 1082 ELSE
803 1 18:2 1088 EXIT(SUBSYSCREATE);
804 1 18:0 1092 END;
805 1 18:0 1110
805 1 18:0 1110 (*I 05:GREET2.TEXT*)

```

See previous page for program description.

```

806 1 34:D 1 (%%P%)PROCEDURE OPENSYSFILES;
807 1 34:0 0 BEGIN
808 1 34:1 0 PROPERMAINDISK;
809 1 34:1 2 (%%I-%)
810 1 34:1 2 RESET(SYSLIST,'APMUTL:APMSYSTEMS');
811 1 34:1 30 (%%I+%)
812 1 34:1 30 IF IORESULT<>0 THEN
813 1 34:2 36 BEGIN
814 1 34:3 36 REWRITE(SYSLIST,'APMUTL:APMSYSTEMS');
815 1 34:3 66 FOR I:=1 TO 10 DO
816 1 34:4 80 BEGIN
817 1 34:5 80 SYSLIST^.NSYSTEM:=I;
818 1 34:5 87 SYSLIST^.SYSTEM:='';
819 1 34:5 97 SEEK(SYSLIST,I);
820 1 34:5 108 PUT(SYSLIST);
821 1 34:5 116 IF EOF(SYSLIST) THEN
822 1 34:6 126 BEGIN
823 1 34:7 126 Writeln('OUT OF DISK SPACE!!!');
824 1 34:7 166 ANYKEY;
825 1 34:7 168 EXIT(SYSTEMFILES);
826 1 34:6 172 END;
827 1 34:4 172 END;
828 1 34:3 182 CLOSE(SYSLIST,LOCK);
829 1 34:3 191 OPENSYSFILES;
830 1 34:3 193 EXIT(OPENSYSFILES);
831 1 34:2 197 END;
832 1 34:1 197 BEGIN
833 1 34:2 197 FOR I:=1 TO 10 DO
834 1 34:3 211 BEGIN
835 1 34:4 211 SEEK(SYSLIST,I);
836 1 34:4 222 GET(SYSLIST);
837 1 34:4 230 NSYSTEM[I]:=SYSLIST^.NSYSTEM;
838 1 34:4 248 SYSTEM[I]:=SYSLIST^.SYSTEM;
839 1 34:3 268 END;
840 1 34:2 278 CLOSE(SYSLIST,LOCK);
841 1 34:1 287 END;
842 1 34:0 287 END;
843 1 34:0 306

```

OPENSYSFILES opens the file containing the list of defined system classes. If such a file does not exist, GOSYSTEMFILES creates one.

```

844 1 35:0 1 (##P)PROCEDURE OPENSUBFILES;
845 1 35:0 0 BEGIN
846 1 35:0 0 (##I-*)
847 1 35:1 0 RESET(SUBSYSLIST,FRAME);
848 1 35:1 11 (##I+*)
849 1 35:1 11 IF IORESULT<>0 THEN
850 1 35:2 17 BEGIN
851 1 35:3 17 REWRITE(SUBSYSLIST,FRAME);
852 1 35:3 30 FOR I:=1 TO 10 DO
853 1 35:4 44 BEGIN
854 1 35:5 44 SUBSYSLIST^.NSUBSYS:=I;
855 1 35:5 51 SUBSYSLIST^.SUBSYS:='';
856 1 35:5 61 SEEK(SUBSYSLIST,I);
857 1 35:5 72 PUT(SUBSYSLIST);
858 1 35:5 80 IF EOF(SUBSYSLIST) THEN
859 1 35:6 90 BEGIN
860 1 35:7 90 WRITELN('OUT OF DISK SPACE!!!');
861 1 35:7 130 ANYKEY;
862 1 35:7 132 EXIT(OPENSUBFILES);
863 1 35:6 136 END;
864 1 35:4 136 END;
865 1 35:3 146 CLOSE(SUBSYSLIST,LOCK);
866 1 35:3 155 RESET(SUBSYSLIST,FRAME);
867 1 35:2 168 END;
868 1 35:1 168 BEGIN
869 1 35:2 168 FOR I:=1 TO 10 DO
870 1 35:3 182 BEGIN
871 1 35:4 182 SEEK(SUBSYSLIST,I);
872 1 35:4 193 GET(SUBSYSLIST);
873 1 35:4 201 NSUBSYS[I]:=SUBSYSLIST^.NSUBSYS;
874 1 35:4 219 SUBSYS[I]:=SUBSYSLIST^.SUBSYS;
875 1 35:3 239 END;
876 1 35:2 249 CLOSE(SUBSYSLIST,LOCK);
877 1 35:1 258 END;
878 1 35:0 258 END;
879 1 35:0 276
880 1 35:0 276

```

OPENSUBFILES opens file containing the names of the subsystems for a given system class and system. If such a file does not exist for the given system class and system, it creates it.

```

881 1 36:D 1 (**P*)PROCEDURE PREPSUBCREATE;
882 1 36:0 0 BEGIN
883 1 36:1 0 FRAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
884 1 36:1 82 OPENSUBFILES;
885 1 36:1 84 SUBSYSCREATE;
886 1 36:0 86 END;
887 1 36:0 98

```

PREPSUBCREATE calls OPENSUBFILES and SUBSYSCREATE as necessary.

```

888 1 37:D 1 ($P$)PROCEDURE SYSCREATE;
889 1 37:0 0 BEGIN
890 1 37:1 0 REPEAT
891 1 37:2 0 I:=0;
892 1 37:2 4 REPEAT
893 1 37:3 4 I:=I+1;
894 1 37:2 12 UNTIL (I=10) OR (SYSTEM[I]='');
895 1 37:2 38 IF I=10 THEN
896 1 37:3 45 BEGIN
897 1 37:4 45 WRITELN('***WARNING SYSTEM IS FULL***');
898 1 37:4 93 ANYKEY;
899 1 37:4 95 EXIT(SYSCREATE);
900 1 37:3 99 END
901 1 37:2 99 ELSE
902 1 37:3 101 BEGIN
903 1 37:4 101 WRITE('What is the name of your new class of systems?');
904 1 37:4 159 SYSTEM[I]:='';
905 1 37:4 177 REPEAT
906 1 37:5 177 READLN(SYSTEM[I]);
907 1 37:5 207 IF SYSTEM[I]='' THEN
908 1 37:6 227 EXIT(SYSCREATE);
909 1 37:5 231 IF LENGTH(SYSTEM[I])<5 THEN
910 1 37:6 250 WRITE('System class name must contain at least 5 characters',
911 1 37:6 314 CHR(13),'Please type a new name:');
912 1 37:5 359 K:=POS(' ',SYSTEM[I]);
913 1 37:5 384 IF (K>0) AND (K<6) THEN
914 1 37:6 397 WRITELN('None of the first five characters of a system class name
can be blank--',chr(13),
'Please type a new name');
915 1 37:6 490 UNTIL(LENGTH(SYSTEM[I])>=5) AND ((K<1) OR (K>5));
916 1 37:4 532 NSYSTEM[I]:=I;
917 1 37:4 563 WRITELN('System class ',SYSTEM[I],' has been added to the list of
system classes');
918 1 37:4 580 WRITELN(' as system number ',NSYSTEM[I]);
919 1 37:4 692 RESET(SYSLIST,'APMUTL:APMSYSTEMS');
920 1 37:4 754 SEEK(SYSLIST,I);
921 1 37:4 784 SYSLIST^.NSYSTEM:=NSYSTEM[I];
922 1 37:4 795 SYSLIST^.SYSTEM:=SYSTEM[I];
923 1 37:4 813 PUT (SYSLIST);
924 1 37:4 833 CLOSE(SYSLIST,LOCK);
925 1 37:4 841

```

SYSCREATE enters system class names into the list of system classes.

```

926 1 37:4 850      CURSYS:=SYSTEM[I];
927 1 37:4 868      NCURSYS:=NSYSTEM[I];
928 1 37:4 885      WRITELN(' ');
929 1 37:4 903      REPEAT
930 1 37:5 903          WRITE('Would you like to define new systems for system class ',
                           SYSTEM[I], '?');
931 1 37:5 1002         HELP:=56;
932 1 37:5 1006         KEY
933 1 37:4 1006         UNTIL (ANS='Y') OR (ANS='N');
934 1 37:4 1021         IF ANS='Y' THEN
935 1 37:5 1028             BEGIN
936 1 37:6 1028                 PREPSPCREATE;
937 1 37:5 1030                 END
938 1 37:4 1030             ELSE
939 1 37:5 1032                 S2;
940 1 37:4 1034                 PREPKEY(33, 'Would you like to develop a new class of systems?');
941 1 37:4 1089                 IF ANS='Y' THEN
942 1 37:5 1096                     OK:=FALSE
943 1 37:4 1096                     ELSE
944 1 37:5 1102                         EXIT(SYSCREATE);
945 1 37:3 1106                     END
946 1 37:1 1106                 UNTIL OK;
947 1 37:0 1111         END;
948 1 37:0 1136

```

See previous page for program description.



```

949 1 5:0 1 (*P*)PROCEDURE SYSTEMFILES;
950 1 5:0 0 BEGIN
951 1 5:1 0 ANSHOLD:= ' ';
952 1 5:1 4 OPENSYSFILES;
953 1 5:1 6 REPEAT
954 1 5:2 6 OVER:=TRUE;
955 1 5:2 10 PAGE(OUTPUT);
956 1 5:2 20 NDATA:=0;
957 1 5:2 24 WRITELN(' I have data for the following classes of human-machine systems:');
958 1 5:2 108 FOR I:=1 TO 10 DO
959 1 5:3 122 BEGIN
960 1 5:4 122 IF SYSTEMC(I)<>' ' THEN
961 1 5:5 142 BEGIN
962 1 5:6 142 WRITELN(' ',SYSTEMC(I),' ',SYSTEMC(I));
963 1 5:6 224 NDATA:=1
964 1 5:5 224 END;
965 1 5:3 228 END;
966 1 5:2 238 WRITELN(' ');
967 1 5:2 256 IF NDATA= 0 THEN
968 1 5:3 263 BEGIN
969 1 5:4 263 I:=0;
970 1 5:4 267 WRITELN(' ... none');
971 1 5:3 297 END
972 1 5:2 297 ELSE
973 1 5:3 299 BEGIN
974 1 5:4 299 REPEAT
975 1 5:5 299 KEYNPREP(33,'Which system class would you like to analyze (type 0
for none of the above)?');
976 1 5:5 381 IF I=999 THEN
977 1 5:6 390 BEGIN
978 1 5:7 390 SYSTEMFILES;
979 1 5:7 392 EXIT(SYSTEMFILES);
980 1 5:6 394 END;
981 1 5:5 396 IF (I<0) OR (I>10) THEN
982 1 5:6 409 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
983 1 5:4 468 UNTIL (I>=0) AND (I<=10);
984 1 5:4 481 IF I<>0 THEN
985 1 5:5 488 BEGIN
986 1 5:6 488 IF SYSTEMC(I)= ' ' THEN
987 1 5:7 508 BEGIN

```

SYSTEMFILES displays the names of defined system classes and determines which one the analyst wishes to analyze.

988	1	5:8	508	WRITELN(NSYSTEM[I], ' DOES NOT EXIST AT PRESENT');
989	1	5:8	577	WRITELN('PLEASE TRY ANOTHER SYSTEM CLASS');
990	1	5:8	628	ANYKEY;
991	1	5:8	630	OVER:=FALSE;
992	1	5:7	634	END;
993	1	5:5	634	END;
994	1	5:3	634	END;
995	1	5:2	634	IF I=0 THEN
996	1	5:3	641	BEGIN
997	1	5:4	641	PREPKEY(33, 'Would you like to develop a new class of systems?');
998	1	5:4	696	IF ANS='Y' THEN
999	1	5:5	703	BEGIN
1000	1	5:6	703	SYSCREATE;
1001	1	5:6	705	OVER:=FALSE
1002	1	5:5	705	END
1003	1	5:4	709	ELSE
1004	1	5:5	711	BEGIN
1005	1	5:6	711	PREPKEY(2, 'Would you like to stop for now?');
1006	1	5:6	748	IF ANS='Y' THEN
1007	1	5:7	755	QUIT
1008	1	5:6	755	ELSE
1009	1	5:7	759	BEGIN
1010	1	5:8	759	WRITELN('There are no other options--so I will present
				the options again');
1011	1	5:8	842	WRITELN('\$\$\$Please press any key to review the options\$\$\$');
1012	1	5:8	910	(\$R-8)
1013	1	5:8	910	READ(ANS);
1014	1	5:8	921	(\$R+8)
1015	1	5:7	921	END;
1016	1	5:5	921	END;
1017	1	5:4	921	OVER:=FALSE;
1018	1	5:3	925	END;
1019	1	5:1	925	UNTIL OVER;
1020	1	5:1	930	CURSYS:=SYSTEM[I];
1021	1	5:1	948	NCURSYS:=NSYSTEM[I];
1022	1	5:0	965	END;
1023	1	5:0	994	

See previous page for program description.

```

1024 1 15:0 1 (**P*)PROCEDURE SUBSYSTEMFILES;
1025 1 15:0 0 BEGIN
1026 1 15:1 0 FRAME:=CONCAT('APHUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
1027 1 15:1 82 OPENSUBFILES;
1028 1 15:1 84 REPEAT
1029 1 15:2 84 OVER:=TRUE;
1030 1 15:2 88 PAGE(OUTPUT);
1031 1 15:2 98 LINE:=CURSP;
1032 1 15:2 105 LLENGTH:=27;
1033 1 15:2 109 WRITE('I have data for the following subsystems of system: ');
1034 1 15:2 173 SHOWALINE;
1035 1 15:2 175 WRITELN(' ');
1036 1 15:2 193 NDATA:=0;
1037 1 15:2 197 FOR I:=1 TO 10 DO
1038 1 15:3 211 BEGIN
1039 1 15:4 211 IF SUBSYS(I)<>' ' THEN
1040 1 15:5 231 BEGIN
1041 1 15:6 231 WRITELN(' ',SUBSYS(I));
1042 1 15:6 313 NDATA:=1;
1043 1 15:5 317 END;
1044 1 15:3 317 END;
1045 1 15:2 327 WRITELN(' ');
1046 1 15:2 345 IF NDATA=0 THEN
1047 1 15:3 352 BEGIN
1048 1 15:4 352 WRITELN(' ... none');
1049 1 15:4 382 S1
1050 1 15:3 382 END
1051 1 15:2 384 ELSE
1052 1 15:3 386 REPEAT
1053 1 15:4 386 REPEAT
1054 1 15:5 386 KEYNPREP(54,'Which subsystem would you like to analyze (type 0 for
none of the above)?');
IF I=999 THEN
BEGIN
SUBSYSTEMFILES;
EXIT(SUBSYSTEMFILES);
END;
IF (I<0) OR (I>10) THEN
WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
UNTIL (I>=0)AND (I<=10);
1055 1 15:5 465
1056 1 15:6 474
1057 1 15:7 474
1058 1 15:7 476
1059 1 15:6 480
1060 1 15:5 480
1061 1 15:6 493
1062 1 15:4 552

```

SUBSYSTEMFILES displays the names of the defined subsystems for a given system.

1063	1	15:4	565	OK:=FALSE;
1064	1	15:4	569	IF I<>0 THEN
1065	1	15:5	576	BEGIN
1066	1	15:6	576	OK:=TRUE;
1067	1	15:6	580	IF SUBSYS[I]=' ' THEN
1068	1	15:7	600	BEGIN
1069	1	15:8	600	WRITELN(NSUBSYS[I], ' DOES NOT EXIST AT PRESENT');
1070	1	15:8	669	WRITELN('PLEASE TRY ANOTHER SYSTEM');
1071	1	15:8	714	OK:=FALSE
1072	1	15:7	714	END;
1073	1	15:5	718	END
1074	1	15:4	718	ELSE
1075	1	15:5	720	BEGIN
1076	1	15:6	720	OK:=TRUE;
1077	1	15:6	724	OVER:=FALSE;
1078	1	15:5	728	END
1079	1	15:3	728	UNTIL OK;
1080	1	15:2	733	IF I=0 THEN
1081	1	15:3	740	SI;
1082	1	15:1	742	UNTIL OVER;
1083	1	15:1	747	CURSUB:=SUBSYS[I];
1084	1	15:1	764	NCURSUB:=NSUBSYS[I];
1085	1	15:0	781	END;
1086	1	15:0	806	

See previous page for program description.

```

1087 1 6:0 1(##P)PROCEDURE S1;
1088 1 6:0 0 BEGIN
1089 1 6:1 0 PREPKEY(54,'Would you like to add subsystems to this system?');
1090 1 6:1 54 IF (ANS='Y') OR (ANS='y') THEN
1091 1 6:2 67 BEGIN
1092 1 6:3 67 SUBSYSCREATE;
1093 1 6:3 69 OVER:=FALSE
1094 1 6:2 69 END
1095 1 6:1 73 ELSE
1096 1 6:2 75 BEGIN
1097 1 6:3 75 PREPKEY(33,'Would you like to process another class of systems?');
1098 1 6:3 132 IF ANS='Y' THEN
1099 1 6:4 139 BEGIN
1100 1 6:5 139 SYSTEMFILES;
1101 1 6:5 141 SPSYSTEMFILES;
1102 1 6:5 143 SUBSYSTEMFILES;
1103 1 6:5 145 PASSNODE^.FLAG1:=0;
1104 1 6:5 153 BRANCHOUT;
1105 1 6:5 155 MENU;
1106 1 6:4 157 END;
1107 1 6:3 157 PREPKEY(2,'Would you like to stop for now?');
1108 1 6:3 194 IF (ANS='Y') OR (ANS='y') THEN
1109 1 6:4 207 QUIT
1110 1 6:3 207 ELSE
1111 1 6:4 211 BEGIN
1112 1 6:5 211 WRITELN('THERE ARE NO MORE OPTIONS--SO I WILL PRESENT OPTIONS LIST AGAIN');
1113 1 6:5 294 WRITELN('***Please press any key to continue***');
1114 1 6:5 352 (##R-#)
1115 1 6:5 352 READ(ANS);
1116 1 6:5 363 (##R+#)
1117 1 6:4 363 END;
1118 1 6:4 363
1119 1 6:3 363 OVER:=FALSE;
1120 1 6:2 367 END;
1121 1 6:0 367 END;
1122 1 6:0 384

```

S1 is a continuation of SUBSYSTEMFILES.

```

1123 1 7:D 1 (%%P%)PROCEDURE S2;
1124 1 7:0 0 BEGIN
1125 1 7:1 0 REPEAT
1126 1 7:2 0 WRITE('You have chosen not to divide system class ',CURSYS,' into systems',
1127 1 7:2 92 chr(13),'Would you like to proceed with applying the model to this system?');
1128 1 7:2 179 HELP:=33;
1129 1 7:2 183 KEY
1130 1 7:1 183 UNTIL (ANS='Y') OR (ANS='N');
1131 1 7:1 198 IF ANS='Y' THEN
1132 1 7:2 205 BEGIN
1133 1 7:3 205 FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
1134 1 7:3 262 OPENSFILES;
1135 1 7:3 264 RESET(SPSYSLIST,FILESPNAME);
1136 1 7:3 277 SEEK(SPSYSLIST,1);
1137 1 7:3 286 SPSYSLIST^.NSPSYS:=1;
1138 1 7:3 291 SPSYSLIST^.SPSYS:=CURSYS;
1139 1 7:3 301 PUT(SPSYSLIST);
1140 1 7:3 309 CLOSE(SPSYSLIST,LOCK);
1141 1 7:3 318 CURSP:=CURSYS;
1142 1 7:3 326 NCURSP:=1;
1143 1 7:3 330 FRAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
1144 1 7:3 412 OPENSUBFILES;
1145 1 7:3 414 RESET(SUBSYSLIST,FRAME);
1146 1 7:3 427 SEEK(SUBSYSLIST,1);
1147 1 7:3 436 SUBSYSLIST^.NSUBSYS:=1;
1148 1 7:3 441 SUBSYSLIST^.SUBSYS:=CURSYS;
1149 1 7:3 451 PUT (SUBSYSLIST);
1150 1 7:3 459 CLOSE(SUBSYSLIST,LOCK);
1151 1 7:3 468 CURSUB:=CURSYS;
1152 1 7:3 475 NCURSUB:=1;
1153 1 7:3 479 PASSNODE^.FLAG1:=0;
1154 1 7:3 487 BRANCHOUT;
1155 1 7:3 489 MENU;
1156 1 7:2 491 END;
1157 1 7:0 491 END;
1158 1 7:0 508
1159 1 7:0 508
1160 1 7:0 508 (%%I 05:GREET2.TEXT%)
1161 1 7:0 508
1161 1 7:0 508 (%%I 05:GREET3.TEXT%)

```

S2 is a continuation of SYSCREATE.

```

1162 1 11:0 1 (**P*)PROCEDURE OPENSPPFILES;
1163 1 11:0 0 BEGIN
1164 1 11:0 0 (*I-*)
1165 1 11:1 0 RESET(SPSYSLIST,FILENAME);
1166 1 11:1 11 (*I+*)
1167 1 11:1 11 IF IORESULT<>0 THEN
1168 1 11:2 17 BEGIN
1169 1 11:3 17 REWRITE(SPSYSLIST,FILENAME);
1170 1 11:3 30 FOR I:=1 TO 10 DO
1171 1 11:4 44 BEGIN
1172 1 11:5 44 SPSYSLIST^.NSPSYS:=I;
1173 1 11:5 51 SPSYSLIST^.SPSYS:='';
1174 1 11:5 61 SEEK(SPSYSLIST,I);
1175 1 11:5 72 PUT(SPSYSLIST);
1176 1 11:5 80 IF EOF(SPSYSLIST) THEN
1177 1 11:6 90 BEGIN
1178 1 11:7 90 WRITELN('OUT OF DISK SPACE!!!');
1179 1 11:7 130 ANYKEY;
1180 1 11:7 132 EXIT(OPENSPPFILES);
1181 1 11:6 136 END;
1182 1 11:4 136 END;
1183 1 11:3 146 CLOSE(SPSYSLIST,LOCK);
1184 1 11:3 155 RESET(SPSYSLIST,FILENAME);
1185 1 11:2 168 END;
1186 1 11:1 168 BEGIN
1187 1 11:2 168 FOR I:=1 TO 10 DO
1188 1 11:3 182 BEGIN
1189 1 11:4 182 SEEK(SPSYSLIST,I);
1190 1 11:4 193 GET(SPSYSLIST);
1191 1 11:4 201 NSPSYS[I]:=SPSYSLIST^.NSPSYS;
1192 1 11:4 219 SPSYS[I]:=SPSYSLIST^.SPSYS;
1193 1 11:3 239 END;
1194 1 11:2 249 CLOSE(SPSYSLIST,LOCK);
1195 1 11:1 258 END;
1196 1 11:0 258 END;
1197 1 11:0 276

```

OPENSPPFILES opens file containing the names of all systems for a particular system class. If such a file does not exist, it creates one.

```

1198 1 13:0 1 (**P*)PROCEDURE SPSYSCREATE;
1199 1 13:0 0 BEGIN
1200 1 13:1 0 REPEAT
1201 1 13:2 0 I:=0;
1202 1 13:2 4 REPEAT
1203 1 13:3 4 I:=I+1;
1204 1 13:2 12 UNTIL(I=10) OR (SPSYS(I)='');
1205 1 13:2 38 IF I=10 THEN
1206 1 13:3 45 BEGIN
1207 1 13:4 45 WRITELN('***WARNING SYST: NO ROOM FOR MORE SPSYSTEMS FOR SYSTEM CLASS',CURSYS);
1208 1 13:4 137 ANYKEY;
1209 1 13:4 139 EXIT(SPSYSCREATE)
1210 1 13:3 143 END
1211 1 13:2 143 ELSE
1212 1 13:3 145 GOSPSYSCREATE;
1213 1 13:1 147 UNTIL OK;
1214 1 13:0 152 END;
1215 1 13:0 168

```

SPSYSCREATE enters new system names into the file of system names for a particular system class.



```

1216 1 12:0 1(##P*)PROCEDURE GOSPSYSCREATE;
1217 1 12:0 0 BEGIN
1218 1 12:1 0 WRITE('What is the name of your system?');
1219 1 12:1 44 SPSYS[I]:='';
1220 1 12:1 62 REPEAT
1221 1 12:2 62 READLN(SPSYS[I]);
1222 1 12:2 92 IF SPSYS[I]='' THEN
1223 1 12:3 112 EXIT(SPSYSCREATE);
1224 1 12:2 116 IF LENGTH(SPSYS[I])<5 THEN
1225 1 12:3 135 WRITE('System name must contain at least 5 letters--',CHR(13),
1226 1 12:3 202 'Please type a new name:');
1227 1 12:2 237 K:=POS(' ',SPSYS[I]);
1228 1 12:2 262 IF (K>0) AND (K<6) THEN
1229 1 12:3 275 WRITE('None of the first five characters of subsystem name can be blank--',chr(13),
1230 1 12:3 363 'Please type a new name:');
1231 1 12:1 398 UNTIL (LENGTH(SPSYS[I])>=5) AND ((K<1) OR (K>5));
1232 1 12:1 429 NSPSYS[I]:=I;
1233 1 12:1 446 WRITELN('System ',SPSYS[I], ' is member number ',NSPSYS[I],chr(13),
1234 1 12:1 550 ' of system class ',CURSYS);
1235 1 12:1 599 RESET(SPSYSLIST,FILESNAME);
1236 1 12:1 612 SEEK(SPSYSLIST,I);
1237 1 12:1 623 SPSYSLIST^.NSPSYS:=NSPSYS[I];
1238 1 12:1 641 SPSYSLIST^.SPSYS:=SPSYS[I];
1239 1 12:1 661 PUT (SPSYSLIST);
1240 1 12:1 669 CLOSE(SPSYSLIST,LOCK);
1241 1 12:1 678 WRITELN(' ');
1242 1 12:1 696 REPEAT
1243 1 12:2 696 WRITE ('Would you like to proceed with the analysis of system class ',
1244 1 12:2 768 ' ',CURSYS,',',CHR(13), ' system ',SPSYS[I],'?');
1245 1 12:2 867 HELP:=33;
1246 1 12:2 871 KEY
1247 1 12:1 871 UNTIL (ANS='Y') OR (ANS='N');
1248 1 12:1 886 IF ANS='Y' THEN
1249 1 12:2 893 BEGIN
1250 1 12:3 893 CURSP:=SPSYS[I];
1251 1 12:3 911 NCURSP:=NSPSYS[I];
1252 1 12:3 928 SUBSYSTEMFILES;
1253 1 12:3 930 PASSNODE^.FLAG1:=0;
1254 1 12:3 938 BRANCHOUT;
1255 1 12:3 940 MENU;

```

GOSPSYSCREATE is a continuation of SPSYSCREATE.

```

1256 1 12:2 942      END;
1257 1 12:1 942      REPEAT
1258 1 12:2 942      WRITE('Would you like to add more systems to system class ',CURSYS,'?');
1259 1 12:2 1027     HELP:=56;
1260 1 12:2 1031     KEY
1261 1 12:1 1031     UNTIL (ANS='Y') OR (ANS='N');
1262 1 12:1 1046     OK:=TRUE;
1263 1 12:1 1050     IF ANS='Y' THEN
1264 1 12:2 1057     OK:=FALSE
1265 1 12:1 1057     ELSE
1266 1 12:2 1063     EXIT(SPSYSCREATE);
1267 1 12:0 1067     END;
1268 1 12:0 1086

```

See previous page for program description.

```

1269 1 16:D 1 (**P*)PROCEDURE PRESPCREATE;
1270 1 16:0 0 BEGIN
1271 1 16:1 0 FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
1272 1 16:1 57 OPENSPFILES;
1273 1 16:1 59 SPSYSCREATE;
1274 1 16:0 61 END;
1275 1 16:0 74

```

PRESPCREATE calls OPENSPFILES and SPSYSCREATE as necessary.

```

1276 1 14:0 1 (*SP*)PROCEDURE SPSYSTEMFILES;
1277 1 14:0 0 BEGIN
1278 1 14:1 0 FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
1279 1 14:1 57 OPENSPPFILES;
1280 1 14:1 59 REPEAT
1281 1 14:2 59 OVER:=TRUE;
1282 1 14:2 63 PAGE(OUTPUT);
1283 1 14:2 73 WRITELN('I have data for the following systems of system class: ',CURSYS);
1284 1 14:2 160 NDATA:=0;
1285 1 14:2 164 FOR I:=1 TO 10 DO
1286 1 14:3 178 BEGIN
1287 1 14:4 178 IF SPSYS[I]<>' THEN
1288 1 14:5 198 BEGIN
1289 1 14:6 198 WRITELN(' ',NSPSYS[I],'. ',SPSYS[I]);
1290 1 14:6 280 NDATA:=1;
1291 1 14:5 284 END;
1292 1 14:3 284 END;
1293 1 14:2 294 WRITELN(' ');
1294 1 14:2 312 IF NDATA=0 THEN
1295 1 14:3 319 BEGIN
1296 1 14:4 319 WRITELN(' ... none');
1297 1 14:4 349 SS
1298 1 14:3 349 END
1299 1 14:2 351 ELSE
1300 1 14:3 353 REPEAT
1301 1 14:4 353 REPEAT
1302 1 14:5 353 KEYNPREP(57,'Which system would you like to analyze (type 0 for
none of the above)?');
IF I=999 THEN
BEGIN
SPSYSTEMFILES;
EXIT(SPSYSTEMFILES);
END;
IF (I<0) OR (I>10) THEN
WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
UNTIL (I>=0)AND (I<=10);
OK:=FALSE;
IF I<>0 THEN
BEGIN
OK:=TRUE;

```

SPSYSTEMFILES displays the name of the system files and determines which system the analyst wishes to use.

1315	1	14:6	544	IF SPSYS(I)='' THEN
1316	1	14:7	544	BEGIN
1317	1	14:8	544	WRITELN(NSPSYS(I), ' DOES NOT EXIST AT PRESENT');
1318	1	14:8	633	WRITELN('PLEASE TRY ANOTHER SYSTEM');
1319	1	14:8	678	OK:=FALSE
1320	1	14:7	678	END;
1321	1	14:5	682	END
1322	1	14:4	682	ELSE
1323	1	14:5	684	BEGIN
1324	1	14:6	684	OK:=TRUE;
1325	1	14:6	688	OVER:=FALSE;
1326	1	14:5	692	END
1327	1	14:3	692	UNTIL OK;
1328	1	14:2	697	IF I=0 THEN
1329	1	14:3	704	S5;
1330	1	14:1	706	UNTIL OVER;
1331	1	14:1	711	CURSP:=SPSYS(I);
1332	1	14:1	729	NCURSP:=NSPSYS(I);
1333	1	14:0	746	END;
1334	1	14:0	770	

See previous page for program description.

```

1335 1 8:0 1(##P)PROCEDURE S5;
1336 1 8:0 0 BEGIN
1337 1 8:1 0 PREPKEY(57,'Would you like to add systems to this class of systems?');
1338 1 8:1 61 IF (ANS='Y') OR (ANS='y') THEN
1339 1 8:2 74 BEGIN
1340 1 8:3 74 SPSYSCREATE;
1341 1 8:3 76 OVER:=FALSE
1342 1 8:2 76 END
1343 1 8:1 80 ELSE
1344 1 8:2 82 BEGIN
1345 1 8:3 82 IF NDATA=0 THEN
1346 1 8:4 89 S2;
1347 1 8:3 91 PREPKEY(57,'Would you like to process another class of systems?');
1348 1 8:3 148 IF ANS='Y' THEN
1349 1 8:4 155 BEGIN
1350 1 8:5 155 SYSTEMFILES;
1351 1 8:5 157 SPSYSTEMFILES;
1352 1 8:5 159 PASSNODE^.FLAG1:=0;
1353 1 8:5 167 BRANCHOUT;
1354 1 8:5 169 MENU;
1355 1 8:4 171 END;
1356 1 8:3 171 PREPKEY(2,'Would you like to stop for now?');
1357 1 8:3 208 IF (ANS='Y') OR (ANS='y') THEN
1358 1 8:4 221 QUIT
1359 1 8:3 221 ELSE
1360 1 8:4 225 BEGIN
1361 1 8:5 225 WRITELN('THERE ARE NO MORE OPTIONS--SO I WILL PRESENT OPTIONS LIST AGAIN');
1362 1 8:5 308 WRITELN('##Please press any key to continue##');
1363 1 8:5 366 ($R-$)
1364 1 8:5 366 READ(ANS);
1365 1 8:5 377 ($R+$)
1366 1 8:4 377 END;
1367 1 8:4 377
1368 1 8:3 377 OVER:=FALSE;
1369 1 8:2 381 END;
1370 1 8:0 381 END;
1371 1 8:0 398
1372 1 8:0 398
1373 1 8:0 398(## 05:GREET3.TEXT#)
1374 1 8:0 398

```

S5 is a contuation of SPSYSTEMFILES.

```

1375 1 1:0 0 BEGIN
1376 1 1:1 0 BRANCHIN;
1377 1 1:1 111 IF PASSNODE^.FLAG1<>1 THEN
1378 1 1:2 121 BEGIN
1379 1 1:3 121 HELLO;
1380 1 1:3 123 PAGE(OUTPUT);
1381 1 1:3 133 PREPKEY(1,'Would you like instructions (type yes or no, then press the
return key)?');
1382 1 1:3 211 IF ANS='Y' THEN
1383 1 1:4 218 INSTRUCTIONS;
1384 1 1:3 220 SYSTEMFILES;
1385 1 1:3 222 SPSYSTEMFILES;
1386 1 1:3 224 SUBSYSTEMFILES;
1387 1 1:2 226 END;
1388 1 1:1 226 BRANCHOUT;
1389 1 1:1 228 MENU;
1390 1 1:0 230 END.

```

MAINPROGRAM: If cold start 1) displays title page, 2) determines which system class/system/and subsystem analyst wants, 3) determines which analytic procedure analyst wants and 4) branches to appropriate analytic procedure for system class, system and subsystem selected.

### **PERFORMANCE ITEM PROGRAM (PERFITEM)**

The performance item program allows the analyst to edit the performance items (objectives, functional purposes and characteristics), adding items, removing items, rewording items and printing out the items available.



```

1 1 1:D 1 ($L PRINTER:$)
2 1 1:D 1 ($S$)
3 1 1:D 1 PROGRAM Builddatafile;
4 1 1:D 3 ($Program to process the performance items$)
5 1 1:D 3 ($Ronald G. Shapiro V2.0 10/25/82$)
6 1 1:D 3
7 28 1:D 3
8 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
9 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
10 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
11 28 5:D 1 PROCEDURE SWAPON;
12 28 6:D 1 PROCEDURE SWAPOFF;
13 28 6:D 1
14 1 1:D 1 USES CHAINSTUFF;
15 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

16 1 1:D 3 (20P8)CONST
17 1 1:D 3 OBJLBL1='The system must be capable of:';
18 1 1:D 3 OBJLBL2='The system must carry out the following activities:';
19 1 1:D 3 OBJLBL3='The system must produce:';
20 1 1:D 3 OBJLBL4='Performance objectives must be met despite:';
21 1 1:D 3 OBJLBL5='Performance objectives must be met despite:';
22 1 1:D 3
23 1 1:D 3 FPURLBL1='This system capability allows:';
24 1 1:D 3 FPURLBL2='The reasons for carrying out this activity are to:';
25 1 1:D 3 FPURLBL3='This product will be used by the system to:';
26 1 1:D 3 FPURLBL4='System purposes must be satisfied despite:';
27 1 1:D 3 FPURLBL5='System purposes must be satisfied despite:';
28 1 1:D 3
29 1 1:D 3 CHARLBL1='For this purpose, the system must have the potential for:';
30 1 1:D 3 CHARLBL2='The tasks required to satisfy this activity are to:';
31 1 1:D 3 CHARLBL3='To realize that product the system must first produce:';
32 1 1:D 3 CHARLBL4='Performance characteristics must be acceptable despite:';
33 1 1:D 3 CHARLBL5='Performance characteristics must be acceptable despite:';
34 1 1:D 3

```

Constants are defined.

```

35 1 1:D 3 (P*)TYPE
36 1 1:D 3
37 1 1:D 3 PASSFILE=RECORD
38 1 1:D 3 CURSYS,CURSP,CURSUB,PAC:STRING(80);
39 1 1:D 3 NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
40 1 1:D 3 END;
41 1 1:D 3
42 1 1:D 3 DATABASE=RECORD
43 1 1:D 3 NTAXA:ARRAY[1..4] OF INTEGER;
44 1 1:D 3 TAXA:STRING(80);
45 1 1:D 3 END;
46 1 1:D 3
47 1 1:D 3 HELPFILE=RECORD
48 1 1:D 3 LINE:ARRAY[1..10] OF STRING(80);
49 1 1:D 3 END;
50 1 1:D 3

```

PASSFILE passes information about 1) system class [CURSYS, NCURSYS], 2) system [CURSP, NCURSP], 3) subsystem [CURSUB, NCURSUB], and 4) aspect [PAC, NPAC] from one program to another. Flag 1 is used to tell the GREETING program whether to begin with title page or analytic procedure list. Flags 2 and 3 are unused. DATABASE contains the performance items. HELPFILE contains the help commands.

```

51 1 1:D 3 (86P8)VAR
52 1 1:D 3 XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB,ANSWER:STRING(80);
53 1 1:D 331 ANSHOLD,ANS2,ANS:CHAR;
54 1 1:D 334 DONE,OK,OVER,POS,NEG:BOOLEAN;
55 1 1:D 339 I,II,II2,J,K,L,M,N,NCHARAC,NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB:INTEGER;
56 1 1:D 354 IMLINECALL,PC,LLENGTH,NLENGTH,PGE,JHELP,TEMP2,LEAVE,HELP,NSCREEN,NPRINT,NDATA,
NCORELAST:INTEGER;
57 1 1:D 367 TSCR,TEMP,CORELAST,EII:INTEGER(8);
58 1 1:D 379 NAMEHELPPFILE,NAMECOREFILE,NAMEDATAFILE:STRING(24);
59 1 1:D 418 REGLINE,LINE:STRING(80);
60 1 1:D 500 APMDISK:STRING(8);
61 1 1:D 505
62 1 1:D 505 ASPECT:ARRAY[1..5] OF STRING(14);
63 1 1:D 545
64 1 1:D 545 SCRATCH:ARRAY[1..20]OF STRING(80);
65 1 1:D 1365 NSCRATCH:ARRAY[1..20] OF INTEGER;
66 1 1:D 1385
67 1 1:D 1385 CORE:ARRAY[1..300] OF INTEGER(8);
68 1 1:D 2285
69 1 1:D 2285 COREFILE:FILE OF INTEGER(8);
70 1 1:D 2588 DATANODE:FILE OF DATABASE;
71 1 1:D 2933 PASSNODE:FILE OF PASSFILE;
72 1 1:D 3404 HELPER: FILE OF HELPPFILE;
73 1 1:D 4114 PRNT:TEXT;
74 1 1:D 4415

```

These strings, arrays and variables are used by this program.

```

75 1 2:D 1 (#P*)PROCEDURE CORECLOSE;FORWARD;
76 1 3:D 1 PROCEDURE CHARCREATE;FORWARD;
77 1 4:D 1 PROCEDURE CHARACTERISTICS;FORWARD;
78 1 5:D 1 PROCEDURE PCHARCREATE;FORWARD;
79 1 6:D 1 PROCEDURE OBJECTIVES;FORWARD;
80 1 7:D 1 PROCEDURE FCC;FORWARD;
81 1 8:D 1 PROCEDURE INDEX;FORWARD;
82 1 9:D 1 PROCEDURE DELFUN;FORWARD;
83 1 10:D 1 PROCEDURE DELCAR;FORWARD;
84 1 11:D 1 PROCEDURE DISPSCRATCH;FORWARD;
85 1 12:D 1 PROCEDURE OBJCREATE;FORWARD;
86 1 13:D 1 PROCEDURE HELPROUTINE;FORWARD;
87 1 14:D 1 PROCEDURE OBJ7;FORWARD;
88 1 15:D 1 PROCEDURE FPUR;FORWARD;
89 1 15:D 1
89 1 15:D 1 (#I 05:PERFITEN2.TEXT *)
90 1 1:D 1 VAR
91 1 1:D 4415 INDENT:INTEGER;
92 1 1:D 4416 LINEOK:BOOLEAN;
93 1 1:D 4417 LONGLINE:STRING[125];
94 1 1:D 4480
95 1 16:D 1 PROCEDURE KEYN;FORWARD;
96 1 17:D 1 PROCEDURE KEY;FORWARD;
97 1 18:D 1 PROCEDURE BRANCHOUT;FORWARD;
98 1 19:D 1 PROCEDURE REMOVE;FORWARD;
99 1 20:D 1 PROCEDURE PREFIXO;FORWARD;
100 1 21:D 1 PROCEDURE PREFIXF;FORWARD;
101 1 22:D 1 PROCEDURE PREFIXC;FORWARD;
102 1 23:D 1 PROCEDURE ANYKEY;FORWARD;
103 1 23:D 1

```

These procedures are presented later on in this program.

```

104 1 24:D 1 (80P)PROCEDURE KEYNPREP(NLP:INTEGER;MSG:STRING);
105 1 24:0 0 BEGIN
106 1 24:1 0 NLP:=NLP;
107 1 24:1 9 WRITE(MSG);
108 1 24:1 20 KEYN;
109 1 24:0 22 END;
110 1 24:0 34

```

KEYNPREP displays a one line message, then calls KEYN to read a number from the keyboard.

```

111 1 25:D 1 (**P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
112 1 25:0 0 BEGIN
113 1 25:1 0 HELP:=HLP;
114 1 25:1 9 REPEAT
115 1 25:2 9 WRITE(MSG);
116 1 25:2 20 KEY;
117 1 25:1 22 UNTIL(ANS='Y') OR (ANS='N');
118 1 25:0 35 END;
119 1 25:0 50

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

120 1 26:0 1 ($P$)PROCEDURE PROPERDISK;
121 1 26:0 0 BEGIN
122 1 26:1 0 REPEAT
123 1 26:1 0 ($I-8)
124 1 26:2 0 RESET(HELPER,NAMEHELPPFILE);
125 1 26:2 11 ($I+8)
126 1 26:2 11 K:=IORESULT;
127 1 26:2 16 IF K=9 THEN
128 1 26:3 23 BEGIN
129 1 26:4 23 PAGE(OUTPUT);
130 1 26:4 33 WRITELN('Please reinsert your data disk into Drive # 2');
131 1 26:4 98 ANYKEY;
132 1 26:3 100 END;
133 1 26:1 100 UNTIL K<>9;
134 1 26:1 107 CLOSE(HELPER);
135 1 26:0 116 END;
136 1 26:0 130

```

PROPERDISK checks to be certain the appropriate subsystem's disk is in Drive #2.



```

137 1 27:0 1 (##P#)PROCEDURE PRNTHelp;
138 1 27:0 0 BEGIN
139 1 27:1 0 DONE:=FALSE;
140 1 27:1 4 REWRITE(PRNT,'PRINTER:');
141 1 27:1 25 PAGE(PRNT);
142 1 27:1 35 WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
143 1 27:1 97 WRITELN(PRNT,CHR(14),'Brief Help File',chr(13));
144 1 27:1 152 PGE:=2;
145 1 27:1 156 REPEAT
146 1 27:2 156 SEEK(HELPER,PGE);
147 1 27:2 167 GET(HELPER);
148 1 27:2 175 PAGE(PRNT);
149 1 27:2 185 K:=PGE-1;
150 1 27:2 193 WRITELN(PRNT,'
151 1 27:2 245 Page ',K);
152 1 27:2 314 FOR J:=1 TO 10 DO
153 1 27:3 328 WRITELN(PRNT,HELPER^.LINE[J]);
154 1 27:2 368 IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
155 1 27:3 405 DONE:=TRUE;
156 1 27:2 409 PGE:=PGE+1;
157 1 27:1 417 UNTIL(DONE);
158 1 27:1 422 PAGE(PRNT);
159 1 27:1 432 CLOSE(PRNT);
160 1 27:0 441 END;
161 1 27:0 458

```

PRNTHelp prints the HELP file on the printer. It is called by HELPROUTINE.

```

162 1 13:0 1 (##P#)PROCEDURE HELPROUTINE;
163 1 13:0 0 BEGIN
164 1 13:0 0 (##I-#)
165 1 13:1 0 RESET(HELPER,'05:HELP');
166 1 13:1 18 (##I+8)
167 1 13:1 18 I:=IORESULT;
168 1 13:1 23 IF (I<>0) THEN
169 1 13:2 30 BEGIN
170 1 13:3 30 PAGE(OUTPUT);
171 1 13:3 40 WRITELN('UNFORTUNATELY, THE HELP FILE IS NOT AVAILABLE ON YOUR DISK');
172 1 13:3 118 WRITELN(' ');
173 1 13:3 136 WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
174 1 13:3 199 READ(ANS);
175 1 13:3 210 EXIT(HELPROUTINE);
176 1 13:2 214 END;
177 1 13:1 214 I:=0;
178 1 13:1 218 PGE:=HELP+1;
179 1 13:1 226 DONE:=FALSE;
180 1 13:1 230 REPEAT
181 1 13:2 230 SEEK(HELPER,PGE);
182 1 13:2 241 GET(HELPER);
183 1 13:2 249 PAGE(OUTPUT);
184 1 13:2 259 GOTOXY(73,0);
185 1 13:2 264 K:=PGE-1;
186 1 13:2 272 WRITELN('Page ',K);
187 1 13:2 309 GOTOXY(0,0);
188 1 13:2 314 FOR J:=1 TO 10 DO
189 1 13:3 328 WRITELN(HELPER^.LINE[J]);
190 1 13:2 368 IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
191 1 13:3 405 DONE:=TRUE;
192 1 13:2 409 WRITELN(' ');
193 1 13:2 427 WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
194 1 13:2 496 WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW ANOTHER PAGE****');
195 1 13:2 589 WRITE ('****PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE HELP ROUTINE****');
196 1 13:2 664 PGE:=PGE+1;
197 1 13:2 672 (##R-#)
198 1 13:2 672 ANSWER:= ' ';
199 1 13:2 689 READLN(ANSWER);
200 1 13:2 708 page(output);

```

HELPROUTINE displays appropriate help commands when it is called by KEY or KEYN. HELPROUTINE knows which HELP to display because the calling program places the appropriate help page number into HELP. Once the analyst sees the first help message, he/she can ask for other help messages by typing in the page number of the desired help messages. Note that the HELP file is made by editing a series of files (HELP1 . . . HELPN) using the Apple editor. Then, they are processed by the BLOCKHELP program (see Chapter VIII). The HELP file produced by BLOCKHELP is suitable for use with the HELPROUTINE. HELPROUTINE "knows" it has hit the last page of the file because the word "conclusion" appears on the second line of the last page.

201	1	13:2	718	IF ORD(ANSWER[1])=27 THEN
202	1	13:3	726	BEGIN
203	1	13:4	726	CLOSE(HELPER);
204	1	13:4	735	PROPERDISK;
205	1	13:4	737	(#R+#)
206	1	13:4	737	EXIT (HELPROUTINE);
207	1	13:4	741	(#R-#)
208	1	13:3	741	END;
209	1	13:2	741	IF (ANSWER[1]>='0') AND (ANSWER[1]<='9') THEN
210	1	13:3	756	BEGIN
211	1	13:4	756	PGE:=ORD(ANSWER[1])-48;
212	1	13:4	765	IF (ANSWER[2]>='0') AND (ANSWER[2]<='9') THEN
213	1	13:5	780	PGE:=PGE*10 + ORD(ANSWER[2])-48;
214	1	13:4	795	PGE:=PGE+1;
215	1	13:4	803	IF PGE<2 THEN
216	1	13:5	810	PGE:=2;
217	1	13:4	814	DONE:=FALSE;
218	1	13:3	818	END;
219	1	13:1	818	UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
220	1	13:1	837	PAGE(OUTPUT);
221	1	13:1	847	PREPKEY(2,'Would you like to print the help file?');
222	1	13:1	891	IF ANS='Y' THEN
223	1	13:2	898	BEGIN
224	1	13:3	898	KEYNPREP(2,'How many copies? ');
225	1	13:3	921	FOR N:=1 TO 1 DO
226	1	13:4	937	PRNTHLP;
227	1	13:3	949	WRITELN('DONE');
228	1	13:2	973	END;
229	1	13:1	973	CLOSE(HELPER);
230	1	13:1	982	PROPERDISK;
231	1	13:1	984	(#R+#)
232	1	13:0	984	END;
233	1	13:0	1004	

See previous page for program description.

```

234 1 17:0 1 ($P*)PROCEDURE KEY;
235 1 17:0 0 BEGIN
236 1 17:0 0 ($R-8)
237 1 17:1 0 ANSWER:= '
238 1 17:1 24 REPEAT
239 1 17:2 24 READLN(ANSWER);
240 1 17:2 43 ANS:=ANSWER[1];
241 1 17:2 50 IF (ANS<>'Y')AND(ANS<>'N')AND(ANS<>'H')AND(ANS<>'y')AND
242 1 17:2 73 (ANS<>'n')AND(ANS<>'h') THEN
243 1 17:3 87 WRITELN('PLEASE RESPOND YES OR NO!');
244 1 17:2 132 IF ORD(ANS)>90 THEN
245 1 17:3 139 BEGIN
246 1 17:4 139 I12:=ORD(ANS)-32;
247 1 17:4 147 ANS:=CHR(I12);
248 1 17:3 153 END;
249 1 17:1 153 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H');
250 1 17:1 172 ($R+8)
251 1 17:1 172 IF ANS='H' THEN
252 1 17:2 179 HELPROUTINE;
253 1 17:0 181 END;
254 1 17:0 196

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

255 1 16:0 1 ($PR)PROCEDURE KEYN;
256 1 16:0 0 BEGIN
257 1 16:0 0 ($R-*)
258 1 16:1 0 ANSWER:= '
259 1 16:1 25 REPEAT
260 1 16:2 25 REPEAT
261 1 16:3 25 READLN(ANSWER);
262 1 16:3 44 IF LENGTH(ANSWER)=0 THEN
263 1 16:4 52 WRITELN('Please enter the integer again');
264 1 16:2 102 UNTIL LENGTH(ANSWER)<>'0';
265 1 16:2 110 ANS:=ANSWER[1];
266 1 16:2 117 ANS2:=ANSWER[2];
267 1 16:2 124 IF (ANS='H') OR (ANS='h') THEN
268 1 16:3 137 BEGIN
269 1 16:4 137 HELPROUTINE;
270 1 16:4 139 I:=999;
271 1 16:4 145 EXIT(KEYN);
272 1 16:3 149 END;
273 1 16:2 149 II:=ORD(ANS)-48;
274 1 16:2 157 II2:=-1;
275 1 16:2 162 II2:=ORD(ANS2)-48;
276 1 16:2 170 IF (II<0) OR (II>9) THEN
277 1 16:3 183 WRITELN('PLEASE RESPOND WITH AN INTEGER!');
278 1 16:1 234 UNTIL (II>=0)AND (II<10);
279 1 16:1 247 I:=II;
280 1 16:1 253 IF (II2>=0)AND(II2<=9) THEN
281 1 16:2 266 I:=I*10+II2;
282 1 16:2 278 ($R+*)
283 1 16:0 278 END;
284 1 16:0 294

```

KEYN reads a 1 or 2 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 2 characters are typed, only 2 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```

285 1 23:D 1 ($P$)PROCEDURE ANYKEY;
286 1 23:0 0 BEGIN
287 1 23:1 0 WRITELN(' ');
288 1 23:1 10 WRITELN('$$$ Please press any key to continue $$$');
289 1 23:1 78 ($R-8)
290 1 23:1 78 READ(ANS);
291 1 23:1 89 ($R+8)
292 1 23:0 89 END;
293 1 23:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

294 1 18:0 1  (%%P%)PROCEDURE BRANCHOUT;
295 1 18:0 0  BEGIN
296 1 18:0 0  (%%I-8)
297 1 18:1 0  REWRITE(PASSNODE,'PASSTHRU');
298 1 18:1 19 (%%I+8)
299 1 18:1 19 IF(IORRESULT<>0) THEN
300 1 18:2 25 WRITE('SERIOUS ERROR -- NO FILE PASSTHRU AT BRANCHOUT')
301 1 18:1 83 ELSE
302 1 18:2 85 BEGIN
303 1 18:3 85 PASSNODE^.CURSYS:=CURSYS;
304 1 18:3 94 PASSNODE^.CURSP:=CURSP;
305 1 18:3 103 PASSNODE^.CURSUB:=CURSUB;
306 1 18:3 112 PASSNODE^.PAC:=PAC;
307 1 18:3 120 PASSNODE^.NCURSYS:=NCURSYS;
308 1 18:3 130 PASSNODE^.NCURSP:=NCURSP;
309 1 18:3 140 PASSNODE^.NCURSUB:=NCURSUB;
310 1 18:3 150 PASSNODE^.NPAC:=NPAC;
311 1 18:3 160 PASSNODE^.FLAG1:=1;
312 1 18:3 168 PUT(PASSNODE);
313 1 18:3 176 IF EOF(PASSNODE) THEN
314 1 18:4 186 WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
315 1 18:3 246 CLOSE(PASSNODE,LOCK);
316 1 18:2 255 END;
317 1 18:0 255 END;
318 1 18:0 270

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```

319 1 20:0 1  (##P)PROCEDURE BRANCHIN;
320 1 20:0 0  BEGIN
321 1 20:0 0  (##I-0)
322 1 20:1 0  RESET(PASSNODE,'PASSTHRU');
323 1 20:1 19 (##I+0)
324 1 20:1 19 IF IORESULT<>0 THEN
325 1 20:2 25 BEGIN
326 1 20:3 25 REWRITE(PASSNODE,'PASSTHRU');
327 1 20:3 46 PASSNODE^.CURSYS:='';
328 1 20:3 56 PASSNODE^.CURSP:='';
329 1 20:3 66 PASSNODE^.CURSUB:='';
330 1 20:3 76 PASSNODE^.PAC:='';
331 1 20:3 84 PASSNODE^.NCURSYS:=0;
332 1 20:3 92 PASSNODE^.NCURSP:=0;
333 1 20:3 100 PASSNODE^.NCURSUB:=0;
334 1 20:3 108 PASSNODE^.NPAC:=0;
335 1 20:3 116 PUT(PASSNODE);
336 1 20:3 124 IF EOF(PASSNODE) THEN
337 1 20:4 134 WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
338 1 20:3 194 CLOSE(PASSNODE,LOCK);
339 1 20:3 203 RESET(PASSNODE,'PASSTHRU')
340 1 20:2 224 END;
341 1 20:1 224 GET(PASSNODE);
342 1 20:1 232 CURSYS:=PASSNODE^.CURSYS;
343 1 20:1 241 CURSP:=PASSNODE^.CURSP;
344 1 20:1 250 CURSUB:=PASSNODE^.CURSUB;
345 1 20:1 259 PAC:=PASSNODE^.PAC;
346 1 20:1 267 NCURSYS:=PASSNODE^.NCURSYS;
347 1 20:1 276 NCURSP:=PASSNODE^.NCURSP;
348 1 20:1 285 NCURSUB:=PASSNODE^.NCURSUB;
349 1 20:1 294 NPAC:=PASSNODE^.NPAC;
350 1 20:1 303 CLOSE(PASSNODE);
351 1 20:0 312 END;
352 1 20:0 326

```

BRANCHIN gets information from the PASSTHRU file for use by this program.



```

353 1 29:0 1  (%%P%)PROCEDURE INLINE;
354 1 29:0 0  BEGIN
355 1 29:1 0  REPEAT
356 1 29:2 0  READLN(LONGLINE);
357 1 29:2 20  LINEOK:=TRUE;
358 1 29:2 24  M:=LENGTH(LONGLINE);
359 1 29:2 32  IF M>80 THEN
360 1 29:3 39  BEGIN
361 1 29:4 39  WRITELN('**WARNING LINE CONTAINS OVER 80 CHARACTERS**');
362 1 29:4 103 WRITELN(' ');
363 1 29:4 121 WRITELN('DO YOU WISH TO TRUNCATE TO 80 CHARACTERS?');
364 1 29:4 182 REPEAT
365 1 29:5 182 HELP:=39;
366 1 29:5 186 KEY
367 1 29:4 186 UNTIL (ANS='Y') OR (ANS='N');
368 1 29:4 201 IF ANS='N' THEN
369 1 29:5 208 BEGIN
370 1 29:6 208 LINEOK:=FALSE;
371 1 29:6 212 WRITELN('PLEASE RE-ENTER LINE:');
372 1 29:5 253 END
373 1 29:4 253 ELSE
374 1 29:5 255 M:=80;
375 1 29:3 259 END;
376 1 29:1 259 UNTIL LINEOK;
377 1 29:1 264 INLINECALL:=INLINECALL+1;
378 1 29:1 272 IF INLINECALL>25 THEN
379 1 29:2 279 BEGIN
380 1 29:3 279 WRITELN('WARNING: You have entered over 25 new performance items',
381 1 29:3 346 chr(13),' during this session. This is the limit allowed in the',
382 1 29:3 424 chr(13),' demonstration system. To enter more, please Select a',
383 1 29:3 501 chr(13),' Different Analytic Procedure. This will re-initialize',
384 1 29:3 579 chr(13),' the stack pointer and allow you to enter more items!');
385 1 29:3 663 ANYKEY;
386 1 29:2 665 END;
387 1 29:1 665 SCRATCH[1]:=COPY(LONGLINE,1,M);
388 1 29:0 694 END;
389 1 29:0 714

```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```

390 1 30:D 1 ($$P$)PROCEDURE SHOWALINE;
391 1 30:0 0 BEGIN
392 1 30:1 0 NLENGTH:=LENGTH(LINE);
393 1 30:1 8 IF NLENGTH<2 THEN
394 1 30:2 15 EXIT(SHOWALINE);
395 1 30:1 19 WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
396 1 30:2 37 NLENGTH:=NLENGTH-1;
397 1 30:1 47 IF NLENGTH<2 THEN
398 1 30:2 54 EXIT(SHOWALINE);
399 1 30:1 58 IF NLENGTH<=LLENGTH THEN
400 1 30:2 67 BEGIN
401 1 30:3 67 WRITE(LINE);
402 1 30:3 79 EXIT(SHOWALINE);
403 1 30:2 83 END;
404 1 30:1 83 L:=LLENGTH;
405 1 30:1 89 WHILE (LINE[L]<>' ') AND (L>1) DO
406 1 30:2 107 L:=L-1;
407 1 30:1 117 L:=L-1;
408 1 30:1 125 IF L>1 THEN
409 1 30:2 132 BEGIN
410 1 30:3 132 REGLINE:=COPY(LINE,1,L);
411 1 30:3 151 WRITELN(REGLINE);
412 1 30:2 171 END;
413 1 30:1 171 L:=L+2;
414 1 30:1 179 NLENGTH:=NLENGTH-L+1;
415 1 30:1 191 IF NLENGTH<1 THEN
416 1 30:2 198 EXIT(SHOWALINE);
417 1 30:1 202 REGLINE:=COPY(LINE,L,NLENGTH);
418 1 30:1 223 WRITE(' ',REGLINE);
419 1 30:1 253 PC:=PC+1;
420 1 30:0 261 END;
421 1 30:0 278

```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```

422 1 31:D 1 (*9P*)PROCEDURE PRINTTOP;
423 1 31:0 0 BEGIN
424 1 31:1 0 M:=LENGTH(CURSYS);
425 1 31:1 7 IF M>16 THEN
426 1 31:2 14 M:=16;
427 1 31:1 18 LINE:=COPY(CURSYS,1,M);
428 1 31:1 36 WRITE(PRNT,'*',LINE,' Systems');
429 1 31:1 78 M:=16-LENGTH(CURSYS);
430 1 31:1 87 FOR L:=1 TO N DO
431 1 31:2 103 WRITE(PRNT,' ');
432 1 31:1 123 M:=LENGTH(CURSP);
433 1 31:1 130 IF M>16 THEN
434 1 31:2 137 M:=16;
435 1 31:1 141 LINE:=COPY(CURSP,1,M);
436 1 31:1 159 WRITE(PRNT,'*',LINE);
437 1 31:1 181 M:=16-LENGTH(CURSP);
438 1 31:1 190 FOR L:=1 TO N DO
439 1 31:2 206 WRITE(PRNT,' ');
440 1 31:1 226 M:=LENGTH(CURSUB);
441 1 31:1 233 IF M>16 THEN
442 1 31:2 240 M:=16;
443 1 31:1 244 LINE:=COPY(CURSUB,1,M);
444 1 31:1 262 WRITE(PRNT,'*',LINE);
445 1 31:1 284 M:=16-LENGTH(CURSUB);
446 1 31:1 293 FOR L:=1 TO N DO
447 1 31:2 309 WRITE(PRNT,' ');
448 1 31:1 329 WRITELN(PRNT,'*',PAC);
449 1 31:1 359 IF NPRINT>1 THEN
450 1 31:2 366 WRITELN(PRNT,'Objective:[',NORJECTIVE,']',XORJECTIVE);
451 1 31:1 431 IF NPRINT>2 THEN
452 1 31:2 438 WRITELN(PRNT,'Fct1 Prps:[',NFUNPUR,']',XFUNPUR);
453 1 31:1 503 WRITELN(PRNT,' ');
454 1 31:0 521 END;
455 1 31:0 540

```

PRINT-TOP prints the current system class, system, subsystem, etc., on the printer.

```

456 1 32:0 1 ($$P$)PROCEDURE PRINTSCRN;
457 1 32:0 0 BEGIN
458 1 32:1 0 REWRITE(PRNT,'PRINTER:');
459 1 32:1 21 PAGE(PRNT);
460 1 32:1 31 PRINTTOP;
461 1 32:1 33 CLOSE(PRNT);
462 1 32:1 42 CLOSE(OUTPUT);
463 1 32:1 51 REWRITE(OUTPUT,'PRINTER:');
464 1 32:1 72 IF NPRINT=1 THEN
465 1 32:2 79 BEGIN
466 1 32:3 79 WRITE('Objectives--');
467 1 32:3 103 PREFIXO;
468 1 32:2 105 END;
469 1 32:1 105 IF NPRINT=2 THEN
470 1 32:2 112 BEGIN
471 1 32:3 112 WRITE('Functional purposes--');
472 1 32:3 145 PREFIXF;
473 1 32:2 147 END;
474 1 32:1 147 IF NPRINT=3 THEN
475 1 32:2 154 BEGIN
476 1 32:3 154 WRITE('Characteristics--');
477 1 32:3 183 PREFIXC;
478 1 32:2 185 END;
479 1 32:1 185 CLOSE(OUTPUT);
480 1 32:1 194 REWRITE(OUTPUT,'CONSOLE:');
481 1 32:1 215 REWRITE(PRNT,'PRINTER:');
482 1 32:1 236 WRITELN(PRNT,' ');
483 1 32:1 254 FOR K:=1 TO 20 DO
484 1 32:2 268 BEGIN
485 1 32:3 268 IF SCRATCHKJ<>' THEN
486 1 32:4 288 BEGIN
487 1 32:5 288 NDATA:=1;
488 1 32:5 292 WRITELN(PRNT,' ',NSCRATCHKJ,' ',SCRATCHKJ)
489 1 32:4 370 END;
490 1 32:2 370 END;
491 1 32:1 380 IF NDATA=0 THEN
492 1 32:2 387 WRITELN(PRNT,'I have no data at this time!!!!');
493 1 32:1 439 CLOSE(PRNT);
494 1 32:0 448 END;
495 1 32:0 462

```

PRINTSCRN prints the performance items currently being displayed on the screen.

```

496 1 33:D 1  (**P*)PROCEDURE TOPSCREEN;
497 1 33:0 0 BEGIN
498 1 33:1 0 PAGE(OUTPUT);
499 1 33:1 10 M:=LENGTH(CURSYS);
500 1 33:1 17 IF M>16 THEN
501 1 33:2 24 M:=16;
502 1 33:1 28 LINE:=COPY(CURSYS,1,M);
503 1 33:1 46 WRITE('* ',LINE,' Systems');
504 1 33:1 88 GOTOXY(26,0);
505 1 33:1 93 M:=LENGTH(CURSP);
506 1 33:1 100 IF M>16 THEN
507 1 33:2 107 M:=16;
508 1 33:1 111 LINE:=COPY(CURSP,1,M);
509 1 33:1 129 WRITE('* ',LINE);
510 1 33:1 151 GOTOXY(44,0);
511 1 33:1 156 M:=LENGTH(CURSUB);
512 1 33:1 163 IF M>16 THEN
513 1 33:2 170 M:=16;
514 1 33:1 174 LINE:=COPY(CURSUB,1,M);
515 1 33:1 192 WRITELN('* ',LINE);
516 1 33:1 222 GOTOXY(62,0);
517 1 33:1 227 WRITELN('* ',PAC);
518 1 33:1 257 M:=LENGTH(XOBJECTIVE);
519 1 33:1 265 IF M>67 THEN M:=67;
520 1 33:1 276 LINE:=COPY(XOBJECTIVE,1,M);
521 1 33:1 295 IF NSCREEN>1 THEN
522 1 33:2 302 WRITELN('Objective[',NOBJECTIVE,']:',LINE);
523 1 33:1 370 M:=LENGTH(XFUNPUR);
524 1 33:1 378 IF M>67 THEN M:=67;
525 1 33:1 389 LINE:=COPY(XFUNPUR,1,M);
526 1 33:1 408 IF NSCREEN>2 THEN
527 1 33:2 415 WRITELN('Fct1 Prps[',NFUNPUR,']:',LINE);
528 1 33:1 483 WRITELN(' ');
529 1 33:0 501 END;
530 1 33:0 514

```

TOPSCREEN displays the system class, system, subsystem, etc., on the top of the display screen.

```

531 1 34:0 1 (%%P%)PROCEDURE OPENCOREFILE;
532 1 34:0 0 BEGIN
533 1 34:0 0 (%%I-%)
534 1 34:1 0 RESET(COREFILE,NAMECOREFILE);
535 1 34:1 11 (%%I+%);
536 1 34:1 11 I:=IORESULT;
537 1 34:1 16 IF I<>0 THEN
538 1 34:2 23 BEGIN
539 1 34:3 23 REWRITE(COREFILE,NAMECOREFILE);
540 1 34:3 36 FOR I:=1 TO 300 DO
541 1 34:4 52 BEGIN
542 1 34:5 52 CORE[I]:=0;
543 1 34:5 79 COREFILE^:=CORE[I];
544 1 34:5 107 PUT(COREFILE);
545 1 34:5 115 IF EOF(COREFILE) THEN
546 1 34:6 125 BEGIN
547 1 34:7 125 WRITELN('OUT OF DISK SPACE!!!');
548 1 34:7 165 ANYKEY;
549 1 34:7 167 BRANCHOUT;
550 1 34:7 169 SETCHAIN('GREETING');
551 1 34:7 183 EXIT(PROGRAM);
552 1 34:6 187 END;
553 1 34:4 187 END;
554 1 34:3 197 CORELAST:=0;
555 1 34:3 212 NCORELAST:=0;
556 1 34:3 216 COREFILE^:=CORELAST;
557 1 34:3 232 PUT(COREFILE);
558 1 34:3 240 CLOSE(COREFILE,LOCK)
559 1 34:2 249 END
560 1 34:1 249 ELSE
561 1 34:2 251 BEGIN
562 1 34:3 251 FOR I:=1 TO 300 DO
563 1 34:4 267 BEGIN
564 1 34:5 267 GET(COREFILE);
565 1 34:5 275 CORE[I]:=COREFILE^;
566 1 34:4 303 END;
567 1 34:3 313 GET(COREFILE);
568 1 34:3 321 CORELAST:=COREFILE^;
569 1 34:3 337 NCORELAST:=TRUNC(CORELAST);
570 1 34:3 350 CLOSE (COREFILE)
571 1 34:2 359 END;
572 1 34:0 359 END;
573 1 34:0 380

```

OPENCOREFILE reads the index to the performance item file into core.

```

574 1 35:0 1 (##P*)PROCEDURE OPENOBJFILE;
575 1 35:0 0 BEGIN
576 1 35:0 0 (##I-*)
577 1 35:1 0 RESET(DATANODE,NAMEDATAFILE);
578 1 35:1 11 (##I+*)
579 1 35:1 11 IF IDRESULT<>0 THEN
580 1 35:2 17 BEGIN
581 1 35:3 17 WRITELN('Please bear with me while I make room for your ',
582 1 35:3 76 'analysis on the disk');
583 1 35:3 116 REWRITE(DATANODE,NAMEDATAFILE);
584 1 35:3 129 FOR I:=1 TO 4 DO
585 1 35:4 143 DATANODE^.NTAXACI:=0;
586 1 35:3 168 FOR I:=1 TO 300 DO
587 1 35:4 184 BEGIN
588 1 35:5 184 DATANODE^.TAXA:='';
589 1 35:5 194 SEEK(DATANODE,I);
590 1 35:5 205 PUT(DATANODE);
591 1 35:5 213 IF EOF(DATANODE) THEN
592 1 35:6 223 BEGIN
593 1 35:7 223 WRITELN('OUT OF DISK SPACE!!!')
594 1 35:6 263 END;
595 1 35:4 263 END;
596 1 35:3 273 CLOSE(DATANODE,LOCK);
597 1 35:3 282 RESET(DATANODE,NAMEDATAFILE)
598 1 35:2 295 END;
599 1 35:0 295 END;
600 1 35:0 314
601 1 35:0 314
602 1 35:0 314 (##I #5:PERFITEM2.TEXT *)
603 1 35:0 314
604 1 35:0 314

```

OPENOBJFILE creates the performance item file if it does not already exist.

```

605 1 22:D 1 (86P)PROCEDURE PREFIXC;
606 1 22:0 0 BEGIN
607 1 22:1 0 CASE NPAC OF
608 1 22:1 5 1: Writeln(CharLb1);
609 1 22:1 84 2: Writeln(CharLb2);
610 1 22:1 157 3: Writeln(CharLb3);
611 1 22:1 233 4: Writeln(CharLb4);
612 1 22:1 310 5: Writeln(CharLb5);
613 1 22:1 387 END;
614 1 22:0 404 END;
615 1 22:0 420

```

PREFIXC displays sentence prefixes for a characteristic.



```

616 1 21:D 1 (**P*)PROCEDURE PREFIXF;
617 1 21:0 0 BEGIN
618 1 21:1 0 CASE NPAC OF
619 1 21:1 5 1: Writeln(FPURLBL1);
620 1 21:1 57 2: Writeln(FPURLBL2);
621 1 21:1 129 3: Writeln(FPURLBL3);
622 1 21:1 194 4: Writeln(FPURLBL4);
623 1 21:1 258 5: Writeln(FPURLBL5);
624 1 21:1 322 END;
625 1 21:0 340 END;
626 1 21:0 356

```

PREFIXF displays sentence prefixes for a functional purpose.

```

627 1 20:D 1 (*P*)PROCEDURE PREFIXO;
628 1 20:0 0 BEGIN
629 1 20:1 0 CASE NPAC OF
630 1 20:1 5 1: Writeln(OBJLBL1);
631 1 20:1 57 2: Writeln(OBJLBL2);
632 1 20:1 130 3: Writeln(OBJLBL3);
633 1 20:1 176 4: Writeln(OBJLBL4);
634 1 20:1 241 5: Writeln(OBJLBL5);
635 1 20:1 306 END;
636 1 20:0 324 END;
637 1 20:0 340

```

PREFIXO displays sentence prefixes for an objective.

```

638 1 36:0 1 ($$P$)PROCEDURE ASPECTS;
639 1 36:0 0 BEGIN
640 1 36:1 0 REPEAT
641 1 36:2 0 PAGE(OUTPUT);
642 1 36:2 10 WRITELN('You are currently analyzing subsystem ',CURSUB,CHR(13),' of the ',
643 1 36:2 102 CURSYS,' class of systems');
644 1 36:2 150 WRITELN(' ');
645 1 36:2 168 ASPECT[1]='Potentialities';
646 1 36:2 198 ASPECT[2]='Processes';
647 1 36:2 223 ASPECT[3]='Products';
648 1 36:2 247 ASPECT[4]='Environaent';
649 1 36:2 274 ASPECT[5]='Constraints';
650 1 36:2 301 HELP:=8;
651 1 36:2 305 WRITELN('To proceed with the analysis you may examine the following');
652 1 36:2 383 WRITELN(' aspects of performance:');
653 1 36:2 428 FOR I:=1 TO 5 DO
654 1 36:3 442 WRITELN(' ',I,'. ',ASPECT[I]);
655 1 36:2 523 WRITELN(' 0. Select a different analytic procedure');
656 1 36:2 586 WRITELN(' ');
657 1 36:2 604 WRITE('Which aspect of subsystem ',CURSUB,' would you like to analyze?');
658 1 36:2 692 REPEAT
659 1 36:3 692 KEYN;
660 1 36:3 694 IF((I<0) OR (I>5)) AND (I<>999) THEN
661 1 36:4 715 WRITELN('PLEASE SELECT AN INTEGER BETWEEN 0 AND 5');
662 1 36:2 775 UNTIL((I>=0) AND (I<=5)) OR (I=999);
663 1 36:1 796 UNTIL I<>999;
664 1 36:1 805 IF I=0 THEN
665 1 36:2 812 BEGIN
666 1 36:3 812 CORECLOSE;
667 1 36:3 814 BRANCHOUT;
668 1 36:3 816 SETCHAIN('GREETING');
669 1 36:3 830 EXIT (PROGRAM);
670 1 36:2 834 END;
671 1 36:1 834 PAC:=ASPECT[I];
672 1 36:1 852 NFAC:=I;
673 1 36:1 858 WRITELN('You have chosen to analyze the ',PAC,' aspect',CHR(13),' of subsystem ',
674 1 36:1 969 CURSUB,' performance. ');
675 1 36:0 1013 END;
676 1 36:0 1032

```

ASPECTS allows analyst to select the aspect he/she intends to use.

```

677 1 37:D 1 (##P#)PROCEDURE FPUR1;
678 1 37:0 0 BEGIN
679 1 37:1 0 GOTOXY(0,16);
680 1 37:1 5 WRITE(CHR(11));
681 1 37:1 15 REPEAT
682 1 37:2 15 WRITE('Which functional purpose would you like to analyze (Type 0 to
reconsider)?');
683 1 37:2 101 HELP:=15;
684 1 37:2 105 KEYN;
685 1 37:2 107 IF I=999 THEN
686 1 37:3 116 FPUR;
687 1 37:2 118 IF (I<0)OR(I>20)THEN
688 1 37:3 131 WRITELN('PLEASE INPUT AN INTEGER BETWEEN 1 AND 20');
689 1 37:2 191 IF (I>0)AND(I<20)THEN
690 1 37:3 204 IF SCRATCH[I]=' ' THEN
691 1 37:4 224 BEGIN
692 1 37:5 224 WRITELN(I, ' DOES NOT EXIST AT PRESENT');
693 1 37:5 282 WRITELN('PLEASE TRY ANOTHER FUNCTIONAL PURPOSE');
694 1 37:5 339 I:=25;
695 1 37:4 343 END;
696 1 37:1 343 UNTIL (I>=0) AND (I<20);
697 1 37:1 356 IF I>0 THEN
698 1 37:2 363 BEGIN
699 1 37:3 363 MFUNPUR:=NSCRATCH[I];
700 1 37:3 380 XFUNPUR:=SCRATCH[I];
701 1 37:3 398 CHARACTERISTICS;
702 1 37:2 400 END;
703 1 37:0 400 END;
704 1 37:0 416

```

FPUR1 asks the analyst which functional purpose he/she would like to analyze when he/she requests to analyze characteristics.

```

705 1 38:0 1 ($P*)PROCEDURE FUNCCREATE;
706 1 38:0 0 BEGIN
707 1 38:1 0 REPEAT
708 1 38:2 0 NSCREEN:=2;
709 1 38:2 4 TOPSCREEN;
710 1 38:2 6 WRITELN('You have chosen to create a new functional purpose.');
```

711 1 38:2 77 WRITELN(' ');

712 1 38:2 95 INDEX;

713 1 38:2 97 IF I=0 THEN

714 1 38:3 104 EXIT(FUNCCREATE);

715 1 38:2 108 FCC;

716 1 38:1 110 UNTIL OK;

717 1 38:0 115 END;

718 1 38:0 130

FUNCCREATE helps analysts to create new functional purposes by finding an appropriate index for the new purpose.

```

719 1 7:D 1 (80P)PROCEDURE FCC;
720 1 7:0 0 BEGIN
721 1 7:1 0 WRITELN('Please specify (80 additional characters available) the new ',
722 1 7:1 72 'functional purpose ',chr(13),' within the ',PAC,
723 1 7:1 149 ' aspect of the ',CURSUB,' system. ');
724 1 7:1 215 WRITE(CHR(13));
725 1 7:1 225 PREFIX;
726 1 7:1 227 INLINE;
727 1 7:1 229 IF SCRATCH[I]='' THEN
728 1 7:2 249 EXIT(FUNCCREATE);
729 1 7:1 253 NSCRATCH[I]:=I;
730 1 7:1 270 WRITELN('Done');
731 1 7:1 294 DATANODE^.NTAXA[1]:=NPAC;
732 1 7:1 309 DATANODE^.NTAXA[2]:=NOBJECTIVE;
733 1 7:1 324 DATANODE^.NTAXA[3]:=NSCRATCH[I];
734 1 7:1 350 DATANODE^.NTAXA[4]:=0;
735 1 7:1 363 DATANODE^.TAXA:=SCRATCH[I];
736 1 7:1 383 REPEAT
737 1 7:2 383 BEGIN
738 1 7:3 383 J:=TRUNC(CORELAST);
739 1 7:3 396 IF J>=300 THEN
740 1 7:4 405 BEGIN
741 1 7:5 405 WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
742 1 7:5 495 WRITELN('*** ITEMS! ***');
743 1 7:5 573 ANYKEY;
744 1 7:5 575 EXIT(FUNCCREATE);
745 1 7:4 579 END;
746 1 7:3 579 CORELAST:=CORELAST+1;
747 1 7:3 604 J:=J+1;
748 1 7:3 612 NCORELAST:=J;
749 1 7:3 618 EII:=CORE[J] DIV 1000000;
750 1 7:2 675 END;
751 1 7:1 675 UNTIL EII =0;
752 1 7:1 693 TEMP:=NPAC;
753 1 7:1 710 TSCR:=NOBJECTIVE;
754 1 7:1 727 CORE[J]:=TEMP*1000000+TSCR*10000+NSCRATCH[I]*100+0;
755 1 7:1 838 SEEK(DATANODE,J);
756 1 7:1 849 PUT(DATANODE);
757 1 7:1 857 WRITELN('It will be necessary, at some time, to add characteristics to this',
chr(13),'function

```

FCC accepts the new functional purpose and stores it in the performance item data set.

```

758 17:1 987      PREPKEY(17,'Would you like to specify characteristics at this time?');
759 17:1 1048     IF ANS='Y' THEN
760 17:2 1055       BEGIN
761 17:3 1055         XFUNPUR:=SCRATCH(I);
762 17:3 1073         NFUNPUR:=NSCRATCH(I);
763 17:3 1090         PCHARCREATE;
764 17:3 1092         CHARCREATE;
765 17:3 1094         EXIT(FUNCCREATE);
766 17:2 1098       END;
767 17:1 1098     PREPKEY(15,'Would you like to specify more functional purposes for this objective?');
768 17:1 1174     IF ANS='Y' THEN
769 17:2 1181       OK:=FALSE;
770 17:1 1181       ELSE
771 17:2 1187         EXIT(FUNCCREATE);
772 17:0 1191     END;
773 17:0 1208

```

See previous page for program description.

```

774 1 9:0 1 ($P$)PROCEDURE DELFUN;
775 1 9:0 0 BEGIN
776 1 9:1 0 REPEAT
777 1 9:2 0 GOTOXY(0,18);
778 1 9:2 5 WRITE(chr(11),'Which one do you want to remove(Type 0 to reconsider):');
779 1 9:2 81 HELP:=15;
780 1 9:2 85 KEYN;
781 1 9:2 87 IF I=999 THEN
782 1 9:3 96 FPUR;
783 1 9:2 98 IF (I<0) OR (I>20) THEN
784 1 9:3 111 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
785 1 9:1 170 UNTIL (I>=0)AND(I<21);
786 1 9:1 183 IF I=0 THEN
787 1 9:2 190 BEGIN
788 1 9:3 190 EXIT(DELFUN);
789 1 9:2 194 END;
790 1 9:1 194 PREPKEY(15,'Do you really want to remove this functional purpose & assoc
characteristics?');
791 1 9:1 277 IF ANS='N' THEN
792 1 9:2 284 BEGIN
793 1 9:3 284 EXIT(DELFUN)
794 1 9:2 288 END;
795 1 9:1 288 J:=0;
796 1 9:1 292 REPEAT
797 1 9:2 292 TEMP:=NPAC;
798 1 9:2 309 J:=J+1;
799 1 9:2 317 IF (CORE[J] DIV 100)=(TEMP*10000+NOBJECTIVES*100+I) THEN
800 1 9:3 392 REMOVE;
801 1 9:1 394 UNTIL (J=NCORELAST)
802 1 9:0 401 END;
803 1 9:0 420

```

DELFUN asks analyst which functional purpose he/she wishes to remove and removes the functional purpose and its component characteristics.



```

804 1 39:D 1 ($P2)PROCEDURE FPUR4;
805 1 39:0 0 BEGIN
806 1 39:1 0 GOTOXY(0,16);
807 1 39:1 5 WRITELN(CHR(11));
808 1 39:1 23 REPEAT
809 1 39:2 23 WRITELN('You have chosen to reword a functional purpose');
810 1 39:2 89 WRITELN(' ');
811 1 39:2 107 WRITELN('Which one do you want to reword (Type 0 to reconsider)? ');
812 1 39:2 183 HELP:=15;
813 1 39:2 187 KEYN;
814 1 39:2 189 IF I=999 THEN
815 1 39:3 198 FPUR;
816 1 39:2 200 IF (I<0) OR (I>20) THEN
817 1 39:3 213 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
818 1 39:1 272 UNTIL (I>=0) AND (I<21);
819 1 39:1 285 IF I=0 THEN
820 1 39:2 292 EXIT(FPUR4);
821 1 39:1 296 IF SCRATCH[I]='' THEN
822 1 39:2 316 BEGIN
823 1 39:3 316 WRITELN(NSCRATCH[I],' DOES NOT EXIST');
824 1 39:3 374 EXIT(FPUR4);
825 1 39:2 378 END;
826 1 39:1 378 GOTOXY(0,16);
827 1 39:1 383 WRITELN(CHR(11));
828 1 39:1 401 WRITELN('Please reword (80 characters available) the functional purpose');
829 1 39:1 483 WRITE(CHR(13));
830 1 39:1 493 PREFIX;
831 1 39:1 495 INLINE;
832 1 39:1 497 IF SCRATCH[I]='' THEN
833 1 39:2 517 EXIT(FPUR4);
834 1 39:1 521 NSCRATCH[I]:=I;
835 1 39:1 538 DATANODE^.NTAXA[1]:=NFAC;
836 1 39:1 553 DATANODE^.NTAXA[2]:=NORJECTIVE;
837 1 39:1 568 DATANODE^.NTAXA[3]:=NSCRATCH[I];
838 1 39:1 594 DATANODE^.NTAXA[4]:=0;
839 1 39:1 607 DATANODE^.TAXA:=SCRATCH[I];
840 1 39:1 627 TEMP:=NFAC;
841 1 39:1 644 TSCR:=NORJECTIVE;
842 1 39:1 661 TEMP:=TEMP*1000000+TSCR*10000+NSCRATCH[I]*100+0;
843 1 39:1 760 J:=0;
844 1 39:1 764 REPEAT
845 1 39:2 764 J:=J+1;
846 1 39:1 772 UNTIL TEMP=CORE[J];
847 1 39:1 803 WRITELN('OK');
848 1 39:1 825 SEEK(DATANODE,J);
849 1 39:1 836 PUT(DATANODE);
850 1 39:0 844 END;
851 1 39:0 860

```

FPUR4 asks the analyst which functional purpose to reword and it asks them to reword the functional purpose.

```

852 1 40:D 1 ($$P$)PROCEDURE FPUR5;
853 1 40:0 0 BEGIN
854 1 40:1 0 WRITELN('Please be certain that the printer is ON and ONLINE!!!!');
855 1 40:1 76 NPRINT:=2;
856 1 40:1 80 PRINTSCRN;
857 1 40:0 82 END;
858 1 40:0 94

```

FPUR5 calls PRINTSCRN to print the entire contents of the functional purpose display.

```

859 1 41:D 1 (SP$)PROCEDURE FPUR7;
860 1 41:0 0 BEGIN
861 1 41:1 0 ASPECTS;
862 1 41:1 2 OBJECTIVES;
863 1 41:0 4 END;
864 1 41:0 16

```

FPUR7 calls ASPECTS and OBJECTIVES so the analyst can specify a new aspect.

```

865 1 42:0      1 (*$*)PROCEDURE SELECTFPS;
866 1 42:0      0   BEGIN
867 1 42:1      0     GOTOXY(0,16);
868 1 42:1      5     WRITELN(CHR(11));
869 1 42:1      23    WRITE('You may perform any of the following procedures:',chr(13),
870 1 42:1      93 ' 1. Analyze characteristics          2. Specify new functional purposes',chr(13),
871 1 42:1      187 ' 3. Remove a functional purpose        4. Reorder a functional purpose',chr(13),
872 1 42:1      277 ' 5. Print these functional purposes    6. Analyze a different objective',chr(13),
873 1 42:1      369 ' 7. Analyze a different aspect      8. Select a different analytic proc.',chr(13),
874 1 42:1      465 'Please select one: ');
875 1 42:1      496    REPEAT
876 1 42:2      496      HELP:=15;
877 1 42:2      500      KEYN;
878 1 42:2      502      IF I=999 THEN
879 1 42:3      511        FPUR;
880 1 42:2      513      IF (I<1) OR (I>8) THEN
881 1 42:3      526        WRITELN('Please type an integer between 1 and 8');
882 1 42:1      584      UNTIL (I>=1) AND (I<=8);
883 1 42:1      597      GOTOXY(0,16);
884 1 42:1      602      WRITE(CHR(11));
885 1 42:1      612      CASE I OF
886 1 42:1      617        1: FPUR1;
887 1 42:1      621        2: FUNC_CREATE;
888 1 42:1      625        3: DELFUN;
889 1 42:1      629        4: FPUR4;
890 1 42:1      633        5: FPUR5;
891 1 42:1      637        6: OBJECTIVES;
892 1 42:1      641        7: FPUR7;
893 1 42:1      645        8: OBJ7; (*YES, IT IS OK*)
894 1 42:1      649      END;
895 1 42:0      672    END;
896 1 42:0      686

```

SELECTFPS displays analytic options available to the analyst at the bottom of the functional purposes page.

```

897 1 15:0 1 ($$P$)PROCEDURE FPUR;
898 1 15:0 0 BEGIN
899 1 15:1 0 NSCREEN:=2;
900 1 15:1 4 TOPSCREEN;
901 1 15:1 4 WRITE('Functional purposes--');
902 1 15:1 39 PREFIXF;
903 1 15:1 41 FOR J:=1 TO 20 DO
904 1 15:2 55 BEGIN
905 1 15:3 55 SCRATCHCJJ:='';
906 1 15:3 73 NSCRATCHCJJ:=J
907 1 15:2 86 END;
908 1 15:1 100 FOR I:=1 TO NCORELAST DO
909 1 15:2 116 BEGIN
910 1 15:3 116 IF CORE[I] DIV 10000=NPAC*100+NOBJECTIVE THEN
911 1 15:4 165 BEGIN
912 1 15:5 165 IF CORE[I]-CORE[I] DIV 10000 * 10000 <> 0 THEN
913 1 15:6 239 BEGIN
914 1 15:7 239 IF CORE[I] - CORE[I] DIV 100 * 100 = 0 THEN
915 1 15:8 309 BEGIN
916 1 15:9 309 SEEK(DATANODE,I);
917 1 15:9 320 GET (DATANODE);
918 1 15:9 328 J:=DATANODE^.NTAXA[3];
919 1 15:9 343 NSCRATCHCJJ:=DATANODE^.NTAXA[3];
920 1 15:9 369 SCRATCHCJJ:=DATANODE^.TAXA;
921 1 15:8 389 END;
922 1 15:6 389 END;
923 1 15:4 389 END;
924 1 15:2 389 END;
925 1 15:1 399 DISPCATCH;
926 1 15:1 401 IF NDATA=0 THEN
927 1 15:2 408 WRITELN('I do not have any functional purposes for objective number',
928 1 15:2 478 NOBJECTIVE,' at the present time');
929 1 15:1 530 SELECTFPS;
930 1 15:1 532 FPUR;
931 1 15:0 534 END;
932 1 15:0 556

```

FPUR governs the primary display of functional purposes.

```

933 1 5:0 1 (86P)PROCEDURE PCHARCREATE;
934 1 5:0 0 BEGIN
935 1 5:1 0 FOR J:=1 TO 20 DO
936 1 5:2 14 BEGIN
937 1 5:3 14 SCRATCH[J]:='';
938 1 5:3 32 WSCRATCH[J]:=J
939 1 5:2 45 END;
940 1 5:0 59 END;
941 1 5:0 74

```

PCHARCREATE clears the STRATCH array which is used in producing the body of the display for objectives, functional purposes and characteristics.

```

942 1 0:D 1 (00P0)PROCEDURE INDEX;
943 1 0:0 0 BEGIN
944 1 0:1 0 I:=0;
945 1 0:1 4 FOR J:=20 DOWNT0 1 DO
946 1 0:2 10 BEGIN
947 1 0:3 10 IF (SCRATCH(J)='') THEN
948 1 0:4 38 I:=J;
949 1 0:2 44 END;
950 1 0:1 54 IF I=0 THEN
951 1 0:2 61 BEGIN
952 1 0:3 61 WRITELN('All 20 indexes are currently in use!!');
953 1 0:3 118 ANYKEY;
954 1 0:2 120 END;
955 1 0:0 120 END;
956 1 0:0 134 (00I 05:PERFITEN3.TEXT0)

```

INDEX determines whether there are 20 objectives for a given aspect, 20 functional purposes for a given objective or 20 characteristics for a given functional purpose before allowing new performance items to be added. If the maximum are in use, additional items cannot be added.

```

957 1 43:0 1 (80P)PROCEDURE OBJCNODE;
958 1 43:0 0 BEGIN
959 1 43:1 0 DATANODE^.NTAXA[1]:=NPAC;
960 1 43:1 15 DATANODE^.NTAXA[2]:=NSCRATCH[1];
961 1 43:1 41 DATANODE^.NTAXA[3]:=0;
962 1 43:1 54 DATANODE^.NTAXA[4]:=0;
963 1 43:1 67 DATANODE^.TAXA:=SCRATCH[1];
964 1 43:1 87 REPEAT
965 1 43:2 87 BEGIN
966 1 43:3 87 J:=TRUNC(CORELAST);
967 1 43:3 100 IF J>=300 THEN
968 1 43:4 109 BEGIN
969 1 43:5 109 WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
ITEMS! ***');
970 1 43:5 199 WRITELN('*** THUS, THIS ITEM WAS NOT /DED TO DATA SET ***');
971 1 43:5 277 ANYKEY;
972 1 43:5 279 EXIT(OBJCREATE);
973 1 43:4 283 END;
974 1 43:3 283 CORELAST:=CORELAST+1;
975 1 43:3 308 J:=J+1;
976 1 43:3 316 EII:=CORE[J] DIV 1000000;
977 1 43:3 373 NCORELAST:=J;
978 1 43:2 379 END;
979 1 43:1 379 UNTIL EII =0;
980 1 43:1 397 TEMP:=NPAC;
981 1 43:1 414 TSCR:=NSCRATCH[1];
982 1 43:1 442 CORE[J]:=TEMP*1000000+TSCR*10000+0*100+0;
983 1 43:1 540 SEEK(DATANODE,J);
984 1 43:1 551 PUT(DATANODE);
985 1 43:0 559 END;
986 1 43:0 576

```

OBJCNODE adds a new objective to the performance item list.



```

987 1 44:D 1 (##P)PROCEDURE OBJ1;
988 1 44:0 0 BEGIN
989 1 44:1 0 WRITE('Which objective would you like to analyze (type 0 to reconsider)?');
990 1 44:1 77 REPEAT
991 1 44:2 77 HELP:=23;
992 1 44:2 81 KEYN;
993 1 44:2 83 IF I=999 THEN
994 1 44:3 92 OBJECTIVES;
995 1 44:2 94 IF(I<0)OR(I>20)THEN
996 1 44:3 107 WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
997 1 44:2 167 IF I>0 THEN
998 1 44:3 174 IF SCRATCH[I]='' THEN
999 1 44:4 194 BEGIN
1000 1 44:5 194 WRITELN(I,' DOES NOT EXIST AT PRESENT');
1001 1 44:5 252 WRITELN('PLEASE TRY ANOTHER OBJECTIVE');
1002 1 44:5 300 I:=25;
1003 1 44:4 304 END;
1004 1 44:1 304 UNTIL (I>=0) AND (I<20);
1005 1 44:1 317 IF (I>0) AND (I<21) THEN
1006 1 44:2 330 BEGIN
1007 1 44:3 330 NOBJECTIVE:=NSCRATCH[I];
1008 1 44:3 347 XOBJECTIVE:=SCRATCH[I];
1009 1 44:3 365 FPUR;
1010 1 44:2 367 END;
1011 1 44:0 367 END;
1012 1 44:0 384

```

OBJ1 asks analyst which objective he wishes to analyze when he requests to analyze functional purposes.

```

1013 1 12:0 X%%P)PROCEDURE OBJCREATE;
1014 1 12:0 0 BEGIN
1015 1 12:1 0 REPEAT
1016 1 12:2 0 MSCREEN:=1;
1017 1 12:2 4 TOPSCREEN;
1018 1 12:2 6 WRITELN('You have chosen to create a new objective.');
```

WRITELN(' ');

```

1019 1 12:2 68 INDEX;
1020 1 12:2 86 IF I=0 THEN
1021 1 12:2 88 EXIT(OBJCREATE);
1022 1 12:3 95 WRITELN('Please specify (80 additional characters available) the new objective',
1023 1 12:2 99 CHR(13), ' within the ', PAC, ' aspect of the ', CURSUB, ' subsystem');
1024 1 12:2 180 WRITE(CHR(13));
1025 1 12:2 294 PREFIXO;
1026 1 12:2 304 INLINE;
1027 1 12:2 306 IF SCRATCH[I]='' THEN
1028 1 12:2 308 EXIT(OBJCREATE);
1029 1 12:3 328 MSCRATCH[I]:=I;
1030 1 12:2 332 OBJCNODE;
1031 1 12:2 349 WRITELN('OK');
```

WRITELN('It will, at some time, be necessary to add functional purposes and', chr(13),

characteristics to this objective');

```

1033 1 12:2 373 PREPKEY(17, 'Would you like to specify functional purposes at this time?');
1034 1 12:2 461 IF ANS='Y' THEN
1035 1 12:2 516 BEGIN
1036 1 12:2 581 XOBJECTIVE:=SCRATCH[I];
1037 1 12:3 588 MOBJECTIVE:=MSCRATCH[I];
1038 1 12:4 588 PCHARCREATE; (YES, ITS OK);
1039 1 12:4 606 FUNC_CREATE;
1040 1 12:4 623 EXIT(OBJCREATE);
1041 1 12:4 625 END;
1042 1 12:4 627 PREPKEY(23, 'Would you like to specify more objectives?');
1043 1 12:2 631 IF ANS='Y' THEN
1044 1 12:2 679 OK:=FALSE
1045 1 12:2 699 ELSE
1046 1 12:3 686 EXIT(OBJCREATE);
1047 1 12:2 686 UNTIL OK;
1048 1 12:3 692 END;
1049 1 12:1 696
1050 1 12:0 701
1051 1 12:0 716
```

OBJCREATE accepts a new performance item and calls OBJCNODE to add it to the list of performance items.

1052	1	19:0	1	(80P8)PROCEDURE REMOVE;
1053	1	19:0	0	BEGIN
1054	1	19:1	0	DATANODE^.NTAXA[1]:=0;
1055	1	19:1	13	DATANODE^.NTAXA[2]:=0;
1056	1	19:1	26	DATANODE^.NTAXA[3]:=0;
1057	1	19:1	39	DATANODE^.NTAXA[4]:=0;
1058	1	19:1	52	DATANODE^.TAXA:='';
1059	1	19:1	62	SEEK(DATANODE,J);
1060	1	19:1	73	PUT(DATANODE);
1061	1	19:1	81	CORRECTJ:=0;
1062	1	19:0	108	END;
1063	1	19:0	120	

REMOVE removes unwanted performance items from the performance item file.

```

1064 1 45:D 1 (##P)PROCEDURE DELOBJ;
1065 1 45:0 0 BEGIN
1066 1 45:1 0 REPEAT
1067 1 45:2 0 GOTOXY(0,16);
1068 1 45:2 5 WRITE(chr(11),'Which one do you wish to remove (Type 0 to reconsider):');
1069 1 45:2 82 KEYN;
1070 1 45:2 84 IF (I<0) OR (I>20) THEN
1071 1 45:3 97 WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
1072 1 45:1 157 UNTIL (I>=0)AND(I<21);
1073 1 45:1 170 IF I=0 THEN
1074 1 45:2 177 BEGIN
1075 1 45:3 177 EXIT(DELOBJ);
1076 1 45:2 181 END;
1077 1 45:1 181 PREPKEY(23,'Do you really want to remove this objective & component functional
purposes?');
1078 1 45:1 263 IF ANS='N' THEN
1079 1 45:2 270 BEGIN
1080 1 45:3 270 EXIT(DELOBJ)
1081 1 45:2 274 END;
1082 1 45:1 274 J:=0;
1083 1 45:1 278 REPEAT
1084 1 45:2 278 J:=J+1;
1085 1 45:2 286 IF (CORE[J] DIV 10000)=(NPAC*100+I) THEN
1086 1 45:3 335 REMOVE;
1087 1 45:1 337 UNTIL (J=NCORELAST)
1088 1 45:0 344 END;
1089 1 45:0 362

```

DELOBJ asks analyst which objective he/she wishes to remove and calls REMOVE to remove the objective and component functional purposes and characteristics.

```

1090 1 46:0 1 (80P8)PROCEDURE OBJ4;
1091 1 46:0 0 BEGIN
1092 1 46:1 0 GOTOXY(0,16);
1093 1 46:1 5 WRITE(CHR(11));
1094 1 46:1 15 REPEAT;
1095 1 46:2 15 WRITELN('You have chosen to reward an objective.');
```

1096 1 46:2 74 WRITELN(' ');

1097 1 46:2 92 WRITE('Which one do you want to reward (Type 0 to reconsider)? ');

1098 1 46:2 160 KEYN;

1099 1 46:2 162 IF (I<0) OR (I>20) THEN

1100 1 46:3 175 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');

1101 1 46:1 234 UNTIL (I>=0) AND (I<21);

1102 1 46:1 247 GOTOXY(0,16);

1103 1 46:1 252 WRITE(CHR(11));

1104 1 46:1 262 IF I=0 THEN

1105 1 46:2 269 EXIT(OBJ4);

1106 1 46:1 273 IF (SCRATCH[I]='')THEN

1107 1 46:2 293 BEGIN

1108 1 46:3 293 WRITELN(NSCRATCH[I], ' DOES NOT EXIST');

1109 1 46:3 351 EXIT(OBJ4);

1110 1 46:2 355 END;

1111 1 46:1 355 WRITELN('Please reward (80 characters available) this objective.');

1112 1 46:1 430 WRITE(CHR(13));

1113 1 46:1 440 PREFIXO;

1114 1 46:1 442 INLINE;

1115 1 46:1 444 IF SCRATCH[I]=' ' THEN

1116 1 46:2 464 EXIT(OBJCREATE);

1117 1 46:1 468 NSCRATCH[I]:=I;

1118 1 46:1 485 DATANODE^.NTAXA[1]:=NPAC;

1119 1 46:1 500 DATANODE^.NTAXA[2]:=NSCRATCH[I];

1120 1 46:1 526 DATANODE^.NTAXA[3]:=0;

1121 1 46:1 539 DATANODE^.NTAXA[4]:=0;

1122 1 46:1 552 DATANODE^.TAXA:=SCRATCH[I];

1123 1 46:1 572 TEMP:=NPAC;

1124 1 46:1 589 TSCR:=NSCRATCH[I];

1125 1 46:1 617 TEMP:=TEMP\*1000000+TSCR\*10000+0\*100+0;

1126 1 46:1 703 J:=0;

1127 1 46:1 707 REPEAT

1128 1 46:2 707 J:=J+1;

1129 1 46:1 715 UNTIL TEMP=CORE[J];

1130 1 46:1 746 WRITELN('OK');

1131 1 46:1 768 SEEK(DATANODE,J);

1132 1 46:1 779 PUT(DATANODE);

1133 1 46:0 787 END;

1134 1 46:0 804

OBJ4 asks analyst which objective he/she wishes to reward and asks him/her to reward it.

```

1135 1 47:D 1 ($$P$)PROCEDURE OBJ5;
1136 1 47:0 0 BEGIN
1137 1 47:1 0 WRITELN('Please be certain that the printer is ON and ONLINE!!!!');
1138 1 47:1 76 NPRINT:=1;
1139 1 47:1 80 PRINTSCRN;
1140 1 47:0 82 END;
1141 1 47:0 94

```

OBJ5 calls PRINTSCREEN to print the contents of the objectives display screen.

```

1142 1 14:D 1 (SP*)PROCEDURE OBJ7;
1143 1 14:0 0 BEGIN
1144 1 14:1 0 CORECLOSE;
1145 1 14:1 2 BRANCHOUT;
1146 1 14:1 4 SETCHAIN('GREETING');
1147 1 14:1 18 EXIT(PROGRAM);
1148 1 14:0 22 END;
1149 1 14:0 34

```

OBJ7 transfers control to the GREETING program.

```

1150 1 11:0 1 (80P*)PROCEDURE DISPSCRATCH;
1151 1 11:0 0 BEGIN
1152 1 11:1 0 NDATA:=0;
1153 1 11:1 4 PC:=1;
1154 1 11:1 8 FOR K:=1 TO 20 DO
1155 1 11:2 22 BEGIN
1156 1 11:3 22 IF SCRATCH[K]<>' ' THEN
1157 1 11:4 42 BEGIN
1158 1 11:5 42 IF PC>=10 THEN
1159 1 11:6 49 BEGIN
1160 1 11:7 49 PC:=1;
1161 1 11:7 53 ANYKEY;
1162 1 11:7 55 GOTOXY(0,5);
1163 1 11:7 60 WRITE(CHR(11));
1164 1 11:6 70 END;
1165 1 11:5 70 NDATA:=1;
1166 1 11:5 74 LLENGTH:=72;
1167 1 11:5 78 LINE:=SCRATCH[K];
1168 1 11:5 96 WRITE(' ',NSCRATCHCKJ,' ');
1169 1 11:5 148 SHOWALINE;
1170 1 11:5 150 WRITELN(' ');
1171 1 11:5 168 PC:=PC+1;
1172 1 11:4 176 END;
1173 1 11:2 176 END;
1174 1 11:0 186 END;
1175 1 11:0 204

```

DISPSCRATCH displays the performance items in the body of the main displays.



```

1176 1 48:D 1 ($$P$)PROCEDURE SELECTOBJECTIVES;
1177 1 48:0 0 BEGIN
1178 1 48:1 0 GOTOXY(0,16);
1179 1 48:1 5 WRITE(CHR(11));
1180 1 48:1 15 WRITE('You may perform any of the following procedures:',chr(13),
1181 1 48:1 85 ' 1. Analyze functional purposes      2. Specify a new objective',chr(13),
1182 1 48:1 171 ' 3. Remove an objective          4. Reward an objective',chr(13),
1183 1 48:1 253 ' 5. Print these objectives        6. Analyze a different aspect',chr(13),
1184 1 48:1 342 ' 7. Select a different analytic proc. ',chr(13),
1185 1 48:1 404 'Please select one: ');
1186 1 48:1 435 REPEAT
1187 1 48:2 435 HELP:=23;
1188 1 48:2 439 KEYN;
1189 1 48:2 441 IF I=999 THEN
1190 1 48:3 450 OBJECTIVES;
1191 1 48:2 452 IF (I<1) OR (I>7) THEN
1192 1 48:3 465 WRITELN('Please type an integer between 1 and 7');
1193 1 48:1 523 UNTIL (I>=1) AND (I<=7);
1194 1 48:1 536 GOTOXY(0,16);
1195 1 48:1 541 WRITE(CHR(11));
1196 1 48:1 551 CASE 1 OF
1197 1 48:1 556 1: OBJ1;
1198 1 48:1 560 2: OBJCREATE;
1199 1 48:1 564 3: DELOBJ;
1200 1 48:1 568 4: OBJ4;
1201 1 48:1 572 5: OBJ5;
1202 1 48:1 576 6: ASPECTS;
1203 1 48:1 580 7: OBJ7;
1204 1 48:1 584 END;
1205 1 48:0 606 END;
1206 1 48:0 620

```

SELECTOBJECTIVES prints the menu of the analytic processes available at the objectives level on the bottom of the display screen.

```

1207 1 6:0 1 (##)PROCEDURE OBJECTIVES;
1208 1 6:0 0 BEGIN
1209 1 6:1 0 NSCREEN:=1;
1210 1 6:1 4 TOPSCREEN;
1211 1 6:1 6 WRITE('Objectives--');
1212 1 6:1 30 PREFIXO;
1213 1 6:1 32 FOR J:=1 TO 20 DO
1214 1 6:2 46 BEGIN
1215 1 6:3 46 SCRATCH[J]:='';
1216 1 6:3 64 NSCRATCH[J]:=J
1217 1 6:2 77 END;
1218 1 6:1 91 FOR I:=1 TO NCORELAST DO
1219 1 6:2 107 BEGIN
1220 1 6:3 107 IF CORE[I] DIV 1000000=NPAC THEN
1221 1 6:4 168 BEGIN
1222 1 6:5 168 IF CORE[I] - CORE[I] DIV 10000 * 10000 = 0 THEN
1223 1 6:6 242 BEGIN
1224 1 6:7 242 SEEK(DATANODE,I);
1225 1 6:7 253 GET (DATANODE);
1226 1 6:7 261 J:=DATANODE^.NTAXA[2];
1227 1 6:7 276 NSCRATCH[J]:=DATANODE^.NTAXA[2];
1228 1 6:7 302 SCRATCH[J]:=DATANODE^.TAXA;
1229 1 6:6 322 END;
1230 1 6:4 322 END;
1231 1 6:2 322 END;
1232 1 6:1 332 DISPSCRATCH;
1233 1 6:1 334 IF NDATA=0 THEN
1234 1 6:2 341 Writeln('I have no objectives for aspect ',PAC,' at this time');
1235 1 6:1 430 SELECTOBJECTIVES;
1236 1 6:1 432 OBJECTIVES;
1237 1 6:0 434 END;
1238 1 6:0 454
1239 1 6:0 454 (##) 05:PERFITEM3.TEXT#)
1240 1 6:0 454

```

OBJECTIVES governs the overall main display of the objectives.

```

1241 1 49:D 1 (**P*)PROCEDURE CH1;
1242 1 49:0 0 BEGIN
1243 1 49:1 0 NSCREEN:=3;
1244 1 49:1 4 TOPSCREEN;
1245 1 49:1 6 WRITELN('You have chosen to create a new characteristic.');
```

---

```

1246 1 49:1 73 WRITELN(' ');
1247 1 49:1 91 INDEX;
1248 1 49:0 93 END;
1249 1 49:0 106
```

CH1 calls index to be certain that there is room to add the desired characteristics to the list of performance items.

```

1250 1 3:0 1 ($0P$)PROCEDURE CHARCREATE;
1251 1 3:0 0 BEGIN
1252 1 3:1 0 REPEAT
1253 1 3:2 0 CH1;
1254 1 3:2 2 IF I=0 THEN
1255 1 3:3 9 EXIT (CHARCREATE);
1256 1 3:2 13 WRITELN('Please specify (80 additional characters available) the new
characteristic?');
1257 1 3:2 108 WRITE(CHR(13));
1258 1 3:2 118 PREFIXC;
1259 1 3:2 120 INLINE;
1260 1 3:2 122 IF SCRATCH[I]='' THEN
1261 1 3:3 142 EXIT(CHARCREATE);
1262 1 3:2 144 NSCRATCH[I]:=I;
1263 1 3:2 163 WRITELN('OK');
1264 1 3:2 185 DATANODE^.NTAXA[1]:=NPAC;
1265 1 3:2 200 DATANODE^.NTAXA[2]:=NOBJECTIVE;
1266 1 3:2 215 DATANODE^.NTAXA[3]:=NFUNPUR;
1267 1 3:2 230 DATANODE^.NTAXA[4]:=NSCRATCH[I];
1268 1 3:2 256 DATANODE^.TAXA:=SCRATCH[I];
1269 1 3:2 276 REPEAT
1270 1 3:3 276 BEGIN
1271 1 3:4 276 J:=TRUNC(CORELAST);
1272 1 3:4 289 IF J>=300 THEN
1273 1 3:5 298 BEGIN
1274 1 3:6 298 WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
ITEMS! ***');
1275 1 3:6 388 WRITELN('*** THUS, THIS ITEM WAS NOT ADDED TO DATA SET ***');
1276 1 3:6 466 ANYKEY;
1277 1 3:6 468 EXIT(FUNCCREATE);
1278 1 3:5 472 END;
1279 1 3:4 472 CORELAST:=CORELAST+1;
1280 1 3:4 497 J:=J+1;
1281 1 3:4 505 EII:=CORE[J] DIV 1000000;
1282 1 3:4 562 NCORELAST:=J;
1283 1 3:3 568 END;
1284 1 3:2 568 UNTIL EII =0;
1285 1 3:2 586 TEMP:=NPAC;
1286 1 3:2 603 TSCR:=NOBJECTIVE;
1287 1 3:2 620 CORE[J]:=TEMP*1000000+TSCR*10000+NFUNPUR*100+NSCRATCH[I];

```

CHARCREATE is the main routine to handle creating new characteristics.

1288	1	3:2	733	SEEK(DATANODE,J);
1289	1	3:2	744	PUT(DATANODE);
1290	1	3:2	752	PREPKEY(12,'Would you like to specify more characteristics?');
1291	1	3:2	805	IF ANS='Y' THEN
1292	1	3:3	812	OK:=FALSE
1293	1	3:2	812	ELSE
1294	1	3:3	818	EXIT(CHARCREATE);
1295	1	3:1	822	UNTIL OK;
1296	1	3:0	827	END;
1297	1	3:0	846	

See previous page for program description.

```

1298 1 10:D 1 (##P)PROCEDURE DELCAR;
1299 1 10:0 0 BEGIN
1300 1 10:1 0 OVER:=FALSE;
1301 1 10:1 4 REPEAT
1302 1 10:2 4 WRITE('Which one (Type 0 to reconsider):');
1303 1 10:2 49 HELP:=12;
1304 1 10:2 53 KEYN;
1305 1 10:2 55 IF I=999 THEN
1306 1 10:3 64 CHARACTERISTICS;
1307 1 10:2 66 IF (I<0) OR (I>20) THEN
1308 1 10:3 79 WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
1309 1 10:1 139 UNTIL (I>=0)AND(I<21);
1310 1 10:1 152 IF I=0 THEN
1311 1 10:2 159 BEGIN
1312 1 10:3 159 EXIT(DELCAR);
1313 1 10:2 163 END;
1314 1 10:1 163 PREPKEY(12,'Do you really want to remove this characteristic?');
1315 1 10:1 218 J:=0;
1316 1 10:1 222 IF ANS='N' THEN
1317 1 10:2 229 BEGIN
1318 1 10:3 229 EXIT(DELFUN)
1319 1 10:2 233 END
1320 1 10:1 233 ELSE
1321 1 10:2 235 REPEAT
1322 1 10:3 235 TEMP:=NPAC;
1323 1 10:3 252 TSCR:=NOBJECTIVE;
1324 1 10:3 269 J:=J+1;
1325 1 10:3 277 IF CORE[J]=TEMP*1000000+TSCR*10000+NFUNPUR*100+I THEN
1326 1 10:4 382 REMOVE;
1327 1 10:2 384 UNTIL (J=NCORELAST)
1328 1 10:0 391 END;
1329 1 10:0 412

```

DELCAR asks analyst which characteristic he/she wishes to delete and calls REMOVE to actually remove the characteristic.

```

1330 1 50:0 1 (*6P*)PROCEDURE CHAR3;
1331 1 50:0 0 BEGIN
1332 1 50:1 0 GOTOXY(0,16);
1333 1 50:1 5 WRITE(CHR(11));
1334 1 50:1 15 REPEAT
1335 1 50:2 15 WRITELN('You have chosen to reward a characteristic.');
```

```

1336 1 50:2 78 WRITELN(' ');
1337 1 50:2 96 HELP:=12;
1338 1 50:2 100 WRITE('Which one do you want to reward (Type 0 to reconsider)? ');
1339 1 50:2 168 KEYN;
1340 1 50:2 170 IF I=999 THEN
1341 1 50:3 179 CHARACTERISTICS;
1342 1 50:2 181 IF (I<0) OR (I>20) THEN
1343 1 50:3 194 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
1344 1 50:1 253 UNTIL (I>=0) AND (I<21);
1345 1 50:1 266 IF I=0 THEN
1346 1 50:2 273 EXIT(CHAR3);
1347 1 50:1 277 IF SCRATCH[I]=' ' THEN
1348 1 50:2 297 BEGIN
1349 1 50:3 297 WRITELN(NSCRATCH[I], ' DOES NOT EXIST');
1350 1 50:3 355 EXIT(CHAR3);
1351 1 50:2 359 END;
1352 1 50:1 359 GOTOXY(0,16);
1353 1 50:1 364 WRITE(CHR(11));
1354 1 50:1 374 WRITELN('Please reward (80 characters available) this characteristic?');
1355 1 50:1 454 WRITE(CHR(13));
1356 1 50:1 464 PREFIXC;
1357 1 50:1 466 INLINE;
1358 1 50:1 468 IF SCRATCH[I]=' ' THEN
1359 1 50:2 488 EXIT(CHAR3);
1360 1 50:1 492 NSCRATCH[I]:=I;
1361 1 50:1 509 DATANODE^.NTAXA[1]:=NPAC;
1362 1 50:1 524 DATANODE^.NTAXA[2]:=NORJECTIVE;
1363 1 50:1 539 DATANODE^.NTAXA[3]:=NFUNFUR;
1364 1 50:1 554 DATANODE^.NTAXA[4]:=NSCRATCH[I];
1365 1 50:1 580 DATANODE^.TAXA:=SCRATCH[I];
1366 1 50:1 600 TEMP:=NPAC;
1367 1 50:1 617 TSCR:=NORJECTIVE;
1368 1 50:1 634 TEMP:=TEMP*1000000+TSCR*10000+NFUNFUR*100+NSCRATCH[I];
1369 1 50:1 735 J:=0;
1370 1 50:1 739 REPEAT
1371 1 50:2 739 J:=J+1;
1372 1 50:1 747 UNTIL TEMP=CORE[J];
1373 1 50:1 778 WRITELN('OK');
```

```

1374 1 50:1 800 SEEK(DATANODE,J);
1375 1 50:1 811 PUT(DATANODE);
1376 1 50:0 819 END;
1377 1 50:0 836
```

CHAR3 asks analyst which characteristic he/she wishes to reward. It then asks him/her for the new wording of the characteristic.

```

1378 1 51:D 1 (88P8)PROCEDURE CHAR4;
1379 1 51:0 0 BEGIN
1380 1 51:1 0 WRITELN('Please be certain that the printer is ON and ONLINE!!!!');
1381 1 51:1 76 NPRINT:=3;
1382 1 51:1 80 PRINTSCRN;
1383 1 51:0 82 END;
1384 1 51:0 94

```

CHAR4 calls PRINTSCRN to print the main screen for the characteristics level performance items.



```

1385 1 52:D 1 ($P*)PROCEDURE CHAR6;
1386 1 52:0 0 BEGIN
1387 1 52:1 0 OBJECTIVES;
1388 1 52:1 2 FPUR;
1389 1 52:0 4 END;
1390 1 52:0 16

```

CHAR6 enables analyst to specify a different objective (and consequently, a different functional purpose).

```

1391 1 53:D 1 (##P#)PROCEDURE CHAR7;
1392 1 53:0 0 BEGIN
1393 1 53:1 0 ASPECTS;
1394 1 53:1 2 OBJECTIVES;
1395 1 53:1 4 FPUR;
1396 1 53:0 6 END;
1397 1 53:0 18

```

CHAR7 allows analyst to specify a different aspect (and consequently, a different objective and functional purpose).

```

1398 1 54:D      1 ($P*)PROCEDURE SELECTCHARACTERISTICS;
1399 1 54:0      0 BEGIN
1400 1 54:1      0 GOTOXY(0,16);
1401 1 54:1      5 WRITE(CHR(11));
1402 1 54:1      15 WRITE('You may perform any of the following procedures:',chr(13),
1403 1 54:1      85 ' 1. Specify new characteristics      2. Remove a characteristic',chr(13),
1404 1 54:1      171 ' 3. Reward a characteristic      4. Print these characteristics',chr(13),
1405 1 54:1      261 ' 5. Analyze a different func. purp. 6. Analyze a different objective',chr(13),
1406 1 54:1      353 ' 7. Analyze a different aspect      8. Select a different analytic proc.',chr(13),
1407 1 54:1      449 'Please select one: ');
1408 1 54:1      480 REPEAT
1409 1 54:2      480 HELP:=12;
1410 1 54:2      484 KEYN;
1411 1 54:2      486 IF I=999 THEN
1412 1 54:3      495 CHARACTERISTICS;
1413 1 54:2      497 IF (I<1) OR (I>8) THEN
1414 1 54:3      510 WRITELN('Please type an integer between 1 and 8');
1415 1 54:1      568 UNTIL (I>=1) AND (I<=8);
1416 1 54:1      581 GOTOXY(0,16);
1417 1 54:1      586 WRITE(CHR(11));
1418 1 54:1      596 CASE I OF
1419 1 54:1      601 1: CHARCREATE;
1420 1 54:1      605 2: DELCAR;
1421 1 54:1      609 3: CHAR3;
1422 1 54:1      613 4: CHAR4;
1423 1 54:1      617 5: FPUR;
1424 1 54:1      621 6: CHAR6;
1425 1 54:1      625 7: CHAR7;
1426 1 54:1      629 8: OBJ7; (*YES, IT IS OK*)
1427 1 54:1      633 END;
1428 1 54:0      656 END;
1429 1 54:0      670

```

SELECTCHARACTERISTICS displays the menu of procedures to be performed with characteristics level taxa.

```

1430 1 4:0 1 ($$P$)PROCEDURE CHARACTERISTICS;
1431 1 4:0 0 BEGIN
1432 1 4:1 0 NSCREEN:=3;
1433 1 4:1 4 TOPSCREEN;
1434 1 4:1 6 WRITE('Characteristics--');
1435 1 4:1 35 PREFIX;
1436 1 4:1 37 FOR J:=1 TO 20 DO
1437 1 4:2 51 BEGIN
1438 1 4:3 51 SCRATCHEJJ:='';
1439 1 4:3 69 NSCRATCHEJJ:=J
1440 1 4:2 82 END;
1441 1 4:1 96 FOR I:=1 TO NCORELAST DO
1442 1 4:2 112 BEGIN
1443 1 4:3 112 TEMP:=NPAC;
1444 1 4:3 129 IF CORE[I] DIV 100=TEMP*10000+NDOBJECTIVE*100+NFUNPUR THEN
1445 1 4:4 204 BEGIN
1446 1 4:5 204 IF CORE[I] - CORE[I] DIV 100 * 100 <> 0 THEN
1447 1 4:6 274 BEGIN
1448 1 4:7 274 SEEK(DATANODE,I);
1449 1 4:7 285 GET (DATANODE);
1450 1 4:7 293 J:=DATANODE^.NTAXA[4];
1451 1 4:7 308 NSCRATCHEJJ:=DATANODE^.NTAXA[4];
1452 1 4:7 334 SCRATCHEJJ:=DATANODE^.TAXA;
1453 1 4:6 354 END;
1454 1 4:4 354 END;
1455 1 4:2 354 END;
1456 1 4:1 364 DISPSCRATCH;
1457 1 4:1 366 IF NDATA=0 THEN
1458 1 4:2 373 Writeln('No characteristics are available at the present time');
1459 1 4:1 445 SELECTCHARACTERISTICS;
1460 1 4:1 447 CHARACTERISTICS;
1461 1 4:0 449 END;
1462 1 4:0 470

```

CHARACTERISTICS governs the characteristics display functions.

```

1463 1 2:D 1 (**)PROCEDURE CORECLOSE;
1464 1 2:0 0 BEGIN
1465 1 2:1 0 RESET(COREFILE,NAMECOREFILE);
1466 1 2:1 13 FOR I:=1 TO 300 DO
1467 1 2:2 29 BEGIN
1468 1 2:3 29 COREFILE^:=CORE[I];
1469 1 2:3 57 PUT (COREFILE)
1470 1 2:2 65 END;
1471 1 2:1 75 COREFILE^:=CORELAST;
1472 1 2:1 91 PUT(COREFILE);
1473 1 2:1 99 CLOSE(COREFILE);
1474 1 2:0 108 END;
1475 1 2:0 122

```

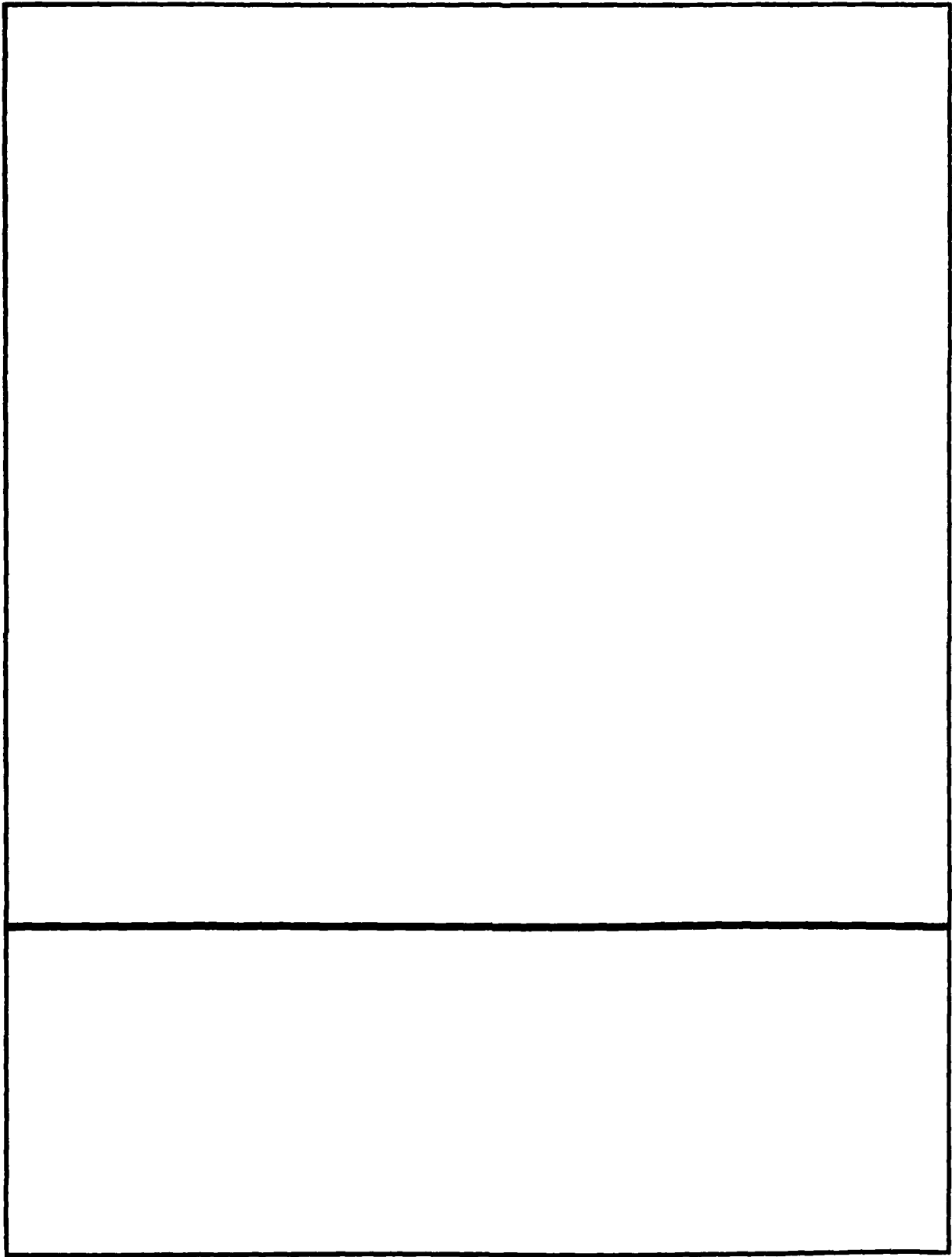
CORECLOSE saves the index to the performance items on disk.

```

1476 1 1:0 0  (##P*)BEGIN
1477 1 1:0 0  (##N-8)
1478 1 1:1 0  INLINECALL:=0;
1479 1 1:1 66  BRANCHIN;
1480 1 1:1 68  APHDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
1481 1 1:1 157  NAMEHELFILE:=CONCAT(APHDSK,'HELP');
1482 1 1:1 193  NAMECOREFILE:=CONCAT(APHDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'CO');
1483 1 1:1 293  NAMEDATAFILE:=CONCAT(APHDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'F1');
1484 1 1:1 393  OPENCOREFILE;
1485 1 1:1 395  ASPECTS;
1486 1 1:1 397  OPENOBJFILE;
1487 1 1:1 399  OBJECTIVES;
1488 1 1:0 401  END.

```

MAIN PROGRAM for analyzing performance items.



### **MEASURES AND ATTRIBUTES PROGRAM (MEASATTR)**

The measurement and attributes program allows the analyst to edit attributes and measures for each performance item, adding, rewording and deleting as appropriate.



```

1 1 1:D 1 ($L PRINTER: #)
2 1 1:D 1 ($S+)
3 1 1:D 1 ($ Program to perform composition of attribute list)
4 1 1:D 1 ($ Ronald G. Shapiro Version 2.0 10/25/82)
5 1 1:D 1 Program Formatattribute;
6 1 1:D 3
7 28 1:D 3
8 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
9 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
10 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
11 28 5:D 1 PROCEDURE SWAPON;
12 28 6:D 1 PROCEDURE SWAPOFF;
13 28 6:D 1
14 1 1:D 1 Uses Chainstuff;
15 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

16 1 1:D 3 (80P2)TYPE
17 1 1:D 3 PASSFILE =RECORD
18 1 1:D 3 CURSYS,CURSP,CURSUB,PAC:STRING[80];
19 1 1:D 3 NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
20 1 1:D 3 END;
21 1 1:D 3
22 1 1:D 3 DATABASE =RECORD
23 1 1:D 3 NTAXA: ARRAY[1..4] OF INTEGER;
24 1 1:D 3 TAXA: STRING[80];
25 1 1:D 3 END;
26 1 1:D 3
27 1 1:D 3 FILEATTRIBUTES =RECORD
28 1 1:D 3 NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
29 1 1:D 3 DESCRIPTOR: STRING[68];
30 1 1:D 3 END;
31 1 1:D 3
32 1 1:D 3 FILEMEASURES =RECORD
33 1 1:D 3 NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
34 1 1:D 3 DESCRIPTOR: STRING[68];
35 1 1:D 3 END;
36 1 1:D 3

```

PASSFILE is used for interprogram communication (see GREETING listing).  
FILEATTRIBUTES contains the attributes.  
FILEMEASURES contains the measures.

```

37 1 1:D 3 (86P8)VAR
38 1 1:D 3 PASSNODE:FILE OF PASSFILE;
39 1 1:D 474 DATANODE:FILE OF DATABASE;
40 1 1:D 819 COREFILE:FILE OF INTEGER(8);
41 1 1:D 1122 ATTRIBUTES:FILE OF FILEATTRIBUTES;
42 1 1:D 1463 ATTRFILE:FILE OF INTEGER(12);
43 1 1:D 1767 MEASURES:FILE OF FILEMEASURES;
44 1 1:D 2108 MEASFILE:FILE OF INTEGER(12);
45 1 1:D 2412
46 1 1:D 2412 CORE:ARRAY[1..300] OF INTEGER(8);
47 1 1:D 3312 ATTRCORE:ARRAY[1..200] OF INTEGER(12);
48 1 1:D 4112 MEASCORE:ARRAY[1..400] OF INTEGER(12);
49 1 1:D 5712
50 1 1:D 5712 SCRATCH:ARRAY [1..20] OF INTEGER;
51 1 1:D 5732 ASPECT:ARRAY[1..5] OF STRING(14);
52 1 1:D 5772 CORE2:ARRAY[1..300] OF INTEGER;
53 1 1:D 6072 ATTRINDEX:ARRAY[1..20] OF INTEGER;
54 1 1:D 6092 MEASINDEX:ARRAY[1..20] OF INTEGER;
55 1 1:D 6112
56 1 1:D 6112 XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING(80);
57 1 1:D 6399 NCURMEASURES,NCURATTRIBUTE,NCURISSUE,NCHARAC,
58 1 1:D 6399 NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
59 1 1:D 6409
60 1 1:D 6409 NAMEATCORE,NAMEATTRIBUTES,NAMENECORE,NAMEMEASURES: STRING(24);
61 1 1:D 6461 CORENAME,DATANAME: STRING(24);
62 1 1:D 6487 APMSDK:STRING(8);
63 1 1:D 6492
64 1 1:D 6492 TEMPL4,TEMP,TEMPL1,TEMPL2,TEMPL3,CORELAST: INTEGER(8);
65 1 1:D 6510 TEMPX,ATTRLAST,MEASLAST:INTEGER(12);
66 1 1:D 6522
67 1 1:D 6522 NODE,INVERSE,HELP,NSCREEN,NPRINT:INTEGER;
68 1 1:D 6527 NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
69 1 1:D 6530 NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;
70 1 1:D 6533
71 1 1:D 6533 INLINECALL,INDENT,LLENGTH,NLENGTH,PC,I,J,K,L,M,N,NATTR,NMEAS,DISPMCOUNT,
DISPCOUNT,COUNT,TEMP2:IN
72 1 1:D 6550
73 1 1:D 6550 REFERENCED,LONGWAY,DONE,OVER,OK,SKIP,NONE:BOOLEAN;
74 1 1:D 6557
75 1 1:D 6557 ANSWER,LINE,REGLINE,LINER:STRING(80);
76 1 1:D 6721
77 1 1:D 6721 ANS,ANSHOLD: CHAR;
78 1 1:D 6723
79 1 1:D 6723 PRNT:TEXT;
80 1 1:D 7024
81 1 2:D 1 PROCEDURE BRANCHIN;FORWARD;
82 1 3:D 1 PROCEDURE BRANCHOUT;FORWARD;
83 1 4:D 1 PROCEDURE ANYKEY;FORWARD;
84 1 4:D 1

```

These strings, arrays and variables are used by this program.

```

85 7 1:0 1 (#0P8)SEGMENT PROCEDURE OPENATTRIBUTESFILE;
86 7 1:0 0 BEGIN
87 7 1:0 0 (#01-8)
88 7 1:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
89 7 1:1 11 (#01+8)
90 7 1:1 11 I:=IORESULT;
91 7 1:1 16 IF I<>0 THEN
92 7 1:2 23 BEGIN
93 7 1:3 23 WRITELN('Please bear with me while I create the attributes file on the disk');
94 7 1:3 109 REWRITE(ATTRIBUTES,NAMEATTRIBUTES);
95 7 1:3 122 FOR I:=1 TO NATTRIBUTES DO
96 7 1:4 138 BEGIN
97 7 1:5 138 SEEK(ATTRIBUTES,I);
98 7 1:5 149 FOR J:=1 TO 6 DO
99 7 1:6 163 ATTRIBUTES^.NDESCRIPTOR(J):=0;
100 7 1:5 188 ATTRIBUTES^.DESCRIPTOR:= '
101 7 1:5 223 PUT(ATTRIBUTES);
102 7 1:5 231 IF (EOF(ATTRIBUTES)) THEN
103 7 1:6 241 BEGIN
104 7 1:7 241 WRITELN('OUT OF DISK SPACE');
105 7 1:7 278 ANYKEY;
106 7 1:7 281 BRANCHOUT;
107 7 1:7 284 SETCHAIN('GREETING');
108 7 1:7 298 EXIT(PROGRAM);
109 7 1:6 302 END;
110 7 1:4 302 END;
111 7 1:3 312 CLOSE(ATTRIBUTES,LOCK);
112 7 1:3 321 OPENATTRIBUTESFILE;
113 7 1:3 323 EXIT(OPENATTRIBUTESFILE);
114 7 1:2 327 END;
115 7 1:1 327 CLOSE(ATTRIBUTES);
116 7 1:0 336 END;
117 7 1:0 356

```

OPENATTRIBUTESFILE creates attributes file if it does not already exist on the disk.

```

118 8 1:0 1 (##P*)SEGMENT PROCEDURE OPENMEASURESFILE;
119 8 1:0 0 BEGIN
120 8 1:0 0 (##I-#)
121 8 1:1 0 RESET(MEASURES,NAMEMEASURES);
122 8 1:1 11 (##I+#)
123 8 1:1 11 I:=IORESULT;
124 8 1:1 16 IF I<>0 THEN
125 8 1:2 23 BEGIN
126 8 1:3 23 WRITELN('Please bear with me while I create the measures file on the disk');
127 8 1:3 107 REWRITE(MEASURES,NAMEMEASURES);
128 8 1:3 120 FOR I:=1 TO NMEASURES DO
129 8 1:4 136 BEGIN
130 8 1:5 136 SEEK(MEASURES,I);
131 8 1:5 147 MEASURES^.DESCRIPTOR:=
132 8 1:5 182 PUT(MEASURES);
133 8 1:5 190 IF (EOF(MEASURES))THEN
134 8 1:6 200 BEGIN
135 8 1:7 200 WRITELN('OUT OF DISK SPACE');
136 8 1:7 237 ANYKEY;
137 8 1:7 240 BRANCHOUT;
138 8 1:7 243 SETCHAIN('GREETING');
139 8 1:7 257 EXIT(PROGRAM);
140 8 1:6 261 END;
141 8 1:4 261 END;
142 8 1:3 271 CLOSE(MEASURES,LOCK);
143 8 1:3 280 OPENMEASURESFILE;
144 8 1:3 282 EXIT(OPENMEASURESFILE);
145 8 1:2 286 END;
146 8 1:1 286 CLOSE(MEASURES);
147 8 1:0 295 END;
148 8 1:0 314

```

OPENMEASURESFILE creates measures file if it does not already exist on the disk.

```

149 9 110 1 (800)SEGMENT PROCEDURE READATTRFILE;
150 9 110 0 BEGIN
151 9 110 0 (801-8)
152 9 111 0 RESET(ATTRFILE,NAMEATCORE);
153 9 111 11 I:=IORESULT;
154 9 110 16 (801+8);
155 9 111 16 IF I<>0 THEN
156 9 112 23 BEGIN
157 9 113 23 REWRITE(ATTRFILE,NAMEATCORE);
158 9 113 36 FOR I:=1 TO NATTRIBUTES DO
159 9 114 52 BEGIN
160 9 115 52 ATTRCORE[I]:=0;
161 9 115 79 ATTRFILE^:=ATTRCORE[I];
162 9 115 107 PUT(ATTRFILE);
163 9 115 115 IF EOF(ATTRFILE) THEN
164 9 116 125 BEGIN
165 9 117 125 WRITELN('OUT OF DISK SPACE');
166 9 116 162 END;
167 9 114 162 END;
168 9 113 172 ATTRLAST:=0;
169 9 113 187 NATTRLAST:=0;
170 9 113 191 ATTRFILE^:=ATTRLAST;
171 9 113 207 PUT(ATTRFILE);
172 9 113 215 CLOSE(ATTRFILE,LOCK);
173 9 112 224 END
174 9 111 224 ELSE
175 9 112 226 BEGIN
176 9 113 226 FOR I:=1 TO NATTRIBUTES DO
177 9 114 242 BEGIN
178 9 115 242 GET(ATTRFILE);
179 9 115 250 ATTRCORE[I]:=ATTRFILE^;
180 9 114 278 END;
181 9 113 288 GET(ATTRFILE);
182 9 113 296 ATTRLAST:=ATTRFILE^;
183 9 113 312 NATTRLAST:=TRUNC(ATTRLAST);
184 9 113 325 CLOSE(ATTRFILE);
185 9 112 334 END;
186 9 110 334 END;
187 9 110 352

```

READATTRFILE loads core with index to attributes file.

```

188 10 1:0 1 (80P8)SEGMENT PROCEDURE READMEASFILE;
189 10 1:0 0 BEGIN
190 10 1:0 0 (80I-8)
191 10 1:1 0 RESET(MEASFILE,NAMEMECORE);
192 10 1:1 11 I:=IORESULT;
193 10 1:0 16 (80I+8);
194 10 1:1 16 IF I<>0 THEN
195 10 1:2 23 BEGIN
196 10 1:3 23 REWRITE(MEASFILE,NAMEMECORE);
197 10 1:3 36 FOR I:=1 TO NMEASURES DO
198 10 1:4 52 BEGIN
199 10 1:5 52 MEASCORE[I]:=0;
200 10 1:5 79 MEASFILE^:=MEASCORE[I];
201 10 1:5 107 PUT(MEASFILE);
202 10 1:5 115 IF EOF(MEASFILE) THEN
203 10 1:6 125 BEGIN
204 10 1:7 125 Writeln('OUT OF DISK SPACE');
205 10 1:7 162 BRANCHOUT;
206 10 1:7 165 SETCHAIN('GREETING');
207 10 1:7 179 EXIT(PROGRAM);
208 10 1:6 183 END;
209 10 1:4 183 END;
210 10 1:3 193 MEASLAST:=0;
211 10 1:3 208 NMEASLAST:=0;
212 10 1:3 212 MEASFILE^:=MEASLAST;
213 10 1:3 228 PUT(MEASFILE);
214 10 1:3 236 CLOSE(MEASFILE,LOCK);
215 10 1:2 245 END
216 10 1:1 245 ELSE
217 10 1:2 247 BEGIN
218 10 1:3 247 FOR I:=1 TO NMEASURES DO
219 10 1:4 263 BEGIN
220 10 1:5 263 GET(MEASFILE);
221 10 1:5 271 MEASCORE[I]:=MEASFILE^;
222 10 1:4 299 END;
223 10 1:3 309 GET(MEASFILE);
224 10 1:3 317 MEASLAST:=MEASFILE^;
225 10 1:3 333 NMEASLAST:=TRUNC(MEASLAST);
226 10 1:3 346 CLOSE(MEASFILE);
227 10 1:2 355 END;
228 10 1:0 355 END;
229 10 1:0 376

```

READMEASFILE loads core with index to measures file.

```

230 11 11D 1 ($P8)SEGMENT PROCEDURE OPENDATAFILE;
231 11 110 0 BEGIN
232 11 110 0 ($I-8)
233 11 111 0 RESET(DATANODE,DATANAME);
234 11 111 11 ($I+8)
235 11 111 11 I:=IORESULT;
236 11 111 16 IF I<>0 THEN
237 11 112 23 BEGIN
238 11 113 23 WRITELN('CREATE DATABASE BEFORE ATTRIBUTES AND MEASURES!');
239 11 113 90 BRANCHOUT;
240 11 113 93 SETCHAIN('GREETING');
241 11 113 107 EXIT(PROGRAM);
242 11 112 111 END;
243 11 110 111 END;
244 11 110 124

```

OPENDATAFILE checks to be sure performance item file exists.



```

245 12 1:D 1 (*P*)SEGMENT PROCEDURE DEFINEASPECTS;
246 12 1:0 0 BEGIN
247 12 1:1 0 ASPECT[1]:='Potentialities';
248 12 1:1 30 ASPECT[2]:='Processes';
249 12 1:1 55 ASPECT[3]:='Products';
250 12 1:1 79 ASPECT[4]:='Environment';
251 12 1:1 106 ASPECT[5]:='Constraints';
252 12 1:0 133 END;
253 12 1:0 146

```

DEFINEASPECTS tells the computer the labels for the aspects file.

```

254 13 1:0 1 ($P$)SEGMENT PROCEDURE READCOREFILE;
255 13 1:0 0 BEGIN
256 13 1:0 0 ($I-$)
257 13 1:1 0 RESET(COREFILE,CORENAME);
258 13 1:1 11 I:=IORESULT;
259 13 1:1 16 ($I+$)
260 13 1:1 16 IF I<>0 THEN
261 13 1:2 23 BEGIN
262 13 1:3 23 WRITELN('COREFILE DOES NOT EXIST');
263 13 1:3 46 ANYKEY;
264 13 1:3 49 BRANCHOUT;
265 13 1:3 72 SETCHAIN('GREETING');
266 13 1:3 86 EXIT(PROGRAM);
267 13 1:2 90 END
268 13 1:1 90 ELSE
269 13 1:2 92 FOR I:=1 TO 300 DO
270 13 1:3 108 BEGIN
271 13 1:4 108 GET(COREFILE);
272 13 1:4 116 CORE[I]:=COREFILE^;
273 13 1:3 144 END;
274 13 1:1 154 GET(COREFILE);
275 13 1:1 162 CORELAST:=COREFILE^;
276 13 1:1 178 NCORELAST:=TRUNC(CORELAST);
277 13 1:1 191 CLOSE(COREFILE)
278 13 1:0 200 END;
279 13 1:0 214

```

READCOREFILE reads index to performance items into core.

```

280 14 1:0 1 (*P*)SEGMENT PROCEDURE SORTCOREFILE;
281 14 1:0 0 BEGIN
282 14 1:1 0 FOR I:=1 TO 300 DO
283 14 1:2 16 CORE2[I]:=I;
284 14 1:1 45 I:=2;
285 14 1:1 49 REPEAT
286 14 1:2 49 IF CORE[I]<CORE[I-1] THEN
287 14 1:3 94 BEGIN
288 14 1:4 94 TEMP:=CORE[I];
289 14 1:4 122 CORE[I]:=CORE[I-1];
290 14 1:4 164 CORE[I-1]:=TEMP;
291 14 1:4 194 TEMP2:=CORE2[I];
292 14 1:4 213 CORE2[I]:=CORE2[I-1];
293 14 1:4 247 CORE2[I-1]:=TEMP2;
294 14 1:4 268 IF I>2 THEN
295 14 1:5 275 I:=I-1;
296 14 1:3 283 END
297 14 1:2 283 ELSE
298 14 1:3 285 I:=I+1;
299 14 1:1 293 UNTIL I>NCORELAST;
300 14 1:0 302 END;
301 14 1:0 320
301 14 1:0 320 (*I 05:UTILITY.TEXT*)
302 14 1:0 320

```

SORTCOREFILE prepares an array CORE2 which lists the location of each performance item in numeric order.

```

303 1 4:D 1 ($P$)PROCEDURE ANYKEY;
304 1 4:0 0 BEGIN
305 1 4:1 0 WRITELN(' ');
306 1 4:1 10 WRITELN('*** Please press any key to continue ***');
307 1 4:1 78 ($R-8)
308 1 4:1 78 READ(ANS);
309 1 4:1 89 ($R+8)
310 1 4:0 89 END;
311 1 4:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

312 1 S:D 1 (##P*)PROCEDURE HELPER;
313 1 S:0 0 BEGIN
314 1 S:1 0 WRITELN('For help please refer to your APM MANUAL.');
```

```

315 1 S:0 61 END;
316 1 S:0 74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```

303 1 4:D 1 (88P)PROCEDURE ANYKEY;
304 1 4:0 0 BEGIN
305 1 4:1 0 WRITELN(' ');
306 1 4:1 18 WRITELN('### Please press any key to continue ###');
307 1 4:1 78 (88R-8)
308 1 4:1 78 READ(ANS);
309 1 4:1 89 (88R+8)
310 1 4:0 89 END;
311 1 4:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

312 1 5:D 1 (##P)PROCEDURE HELPER;
313 1 5:0 0 BEGIN
314 1 5:1 0 WRITELN('For help please refer to your APM MANUAL.');
```

```

315 1 5:0 61 END;
316 1 5:0 74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```

317 1 6:B 1 (88P)PROCEDURE KEYN;
318 1 6:D 1 VAR
319 1 6:D 1 ANSWER: STRING[40];
320 1 6:D 22 II: ARRAY[1..4] OF INTEGER;
321 1 6:D 24 OK: BOOLEAN;
322 1 6:D 27 IIO: INTEGER;
323 1 6:D 28
324 1 6:O 0 BEGIN
325 1 6:O 0 (88R-8)
326 1 6:1 0 REPEAT
327 1 6:2 0 REPEAT
328 1 6:3 0 ANSWER:=
329 1 6:3 27 OK:=TRUE;
330 1 6:3 30 READLN(ANSWER);
331 1 6:3 49 IF LENGTH(ANSWER)=0 THEN
332 1 6:4 57 WRITELN('Please enter the integer again');
333 1 6:2 107 UNTIL LENGTH(ANSWER)<>0;
334 1 6:2 115 IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
335 1 6:3 130 HELPER;
336 1 6:2 132 FOR I:=1 TO 4 DO
337 1 6:3 147 BEGIN
338 1 6:4 147 IIC[1]:=ORD(ANSWER[1])-48;
339 1 6:4 165 IF (IIC[1]<0) OR (IIC[1]>9) THEN
340 1 6:5 192 BEGIN
341 1 6:6 192 IF (I=1) OR (IIC[1]<>(ORD(' ')-48)) THEN
342 1 6:7 214 BEGIN
343 1 6:8 214 OK:=FALSE;
344 1 6:8 217 WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
345 1 6:7 275 END;
346 1 6:5 275 END;
347 1 6:3 275 END;
348 1 6:1 285 UNTIL OK=TRUE;
349 1 6:1 292 IIO:=IIC[1];
350 1 6:1 302 FOR I:=2 TO 4 DO
351 1 6:2 317 BEGIN
352 1 6:3 317 IF (IIC[1]>=0) AND (IIC[1]<=9) THEN
353 1 6:4 344 IIO:=IIO*10+IIC[1];
354 1 6:2 361 END;
355 1 6:2 371 (88R+8)
356 1 6:1 371 I:=IIO;
357 1 6:O 376 END;
358 1 6:O 398

```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.



```

359 1 7:D 1 (88P8)PROCEDURE KEY;
360 1 7:D 1 VAR
361 1 7:D 1 II2:INTEGER;
362 1 7:0 0 BEGIN
363 1 7:0 0 (88R-8)
364 1 7:1 0 ANSWER:= '
365 1 7:1 27 REPEAT
366 1 7:2 27 READLN(ANSWER);
367 1 7:2 47 ANS:=ANSWER[1];
368 1 7:2 55 IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
369 1 7:2 78 (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<27) THEN
370 1 7:3 98 WRITELN('PLEASE RESPOND YES OR NO!');
371 1 7:2 143 IF (ORD(ANS)>90) THEN
372 1 7:3 150 BEGIN
373 1 7:4 150 II2:=ORD(ANS)-32;
374 1 7:4 157 ANS:=CHR(II2);
375 1 7:3 161 END;
376 1 7:1 161 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
377 1 7:1 186 (88R+8)
378 1 7:1 186 IF ANS='H' THEN
379 1 7:2 193 HELPER;
380 1 7:0 195 END;
381 1 7:0 210

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

382 1 0:D 1 (80P)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
383 1 0:0 0 BEGIN
384 1 0:1 0 HLP:=HLP;
385 1 0:1 9 REPEAT
386 1 0:2 9 WRITE(MSG);
387 1 0:2 20 KEY;
388 1 0:1 22 UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
389 1 0:0 41 END;
390 1 0:0 56

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

391 1 9:D 1 (80P)PROCEDURE INLINE;
392 1 9:D 1 VAR
393 1 9:D 1 LONGLINE:STRING(125);
394 1 9:D 64 LINEOK:BOOLEAN;
395 1 9:D 65
396 1 9:0 0 BEGIN
397 1 9:1 0 REPEAT
398 1 9:2 0 READLN(LONGLINE);
399 1 9:2 19 LINEOK:=TRUE;
400 1 9:2 22 M:=LENGTH(LONGLINE);
401 1 9:2 29 IF M>68 THEN
402 1 9:3 34 BEGIN
403 1 9:4 36 WRITELN('WARNING LINE CONTAINS OVER 68 CHARACTERS');
404 1 9:4 100 WRITELN(' ');
405 1 9:4 118 PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 68 CHARACTERS? ');
406 1 9:4 166 IF ANS='N' THEN
407 1 9:5 173 BEGIN
408 1 9:6 173 LINEOK:=FALSE;
409 1 9:6 176 WRITELN('PLEASE TYPE LINE AGAIN: ');
410 1 9:5 220 END
411 1 9:4 220 ELSE
412 1 9:5 222 M:=68;
413 1 9:3 226 END;
414 1 9:1 226 UNTIL LINEOK;
415 1 9:1 230 INLINECALL:=INLINECALL+1;
416 1 9:1 238 IF INLINECALL>25 THEN
417 1 9:2 245 BEGIN
418 1 9:3 245 WRITELN('WARNING--You have typed in over 25 new attributes and/or',
419 1 9:3 313 chr(13),' measures--the limit for the demonstration. Please select',
420 1 9:3 394 chr(13),' a different analytic procedure before entering more data',
421 1 9:3 474 chr(13),' --or risk losing everything you have done today!');
422 1 9:3 554 ANYKEY;
423 1 9:2 556 END;
424 1 9:1 556 LINE:=COPY(LONGLINE,1,M);
425 1 9:0 574 END;
426 1 9:0 592

```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```

427 1 10:0 1 (88P)PROCEDURE SHOWALINE;
428 1 10:0 0 BEGIN
429 1 10:1 0 NLENGTH:=LENGTH(LINE);
430 1 10:1 8 IF NLENGTH<2 THEN
431 1 10:2 15 EXIT(SHOWALINE);
432 1 10:1 19 WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
433 1 10:2 37 NLENGTH:=NLENGTH-1;
434 1 10:1 47 IF NLENGTH<2 THEN
435 1 10:2 54 EXIT(SHOWALINE);
436 1 10:1 58 IF NLENGTH<=LENGTH THEN
437 1 10:2 67 BEGIN
438 1 10:3 67 WRITE(LINE);
439 1 10:3 79 EXIT(SHOWALINE);
440 1 10:2 83 END;
441 1 10:1 83 L:=NLENGTH;
442 1 10:1 89 WHILE (LINE[L]<>' ') AND (L>1) DO
443 1 10:2 107 L:=L-1;
444 1 10:1 117 L:=L-1;
445 1 10:1 125 IF L>0 THEN
446 1 10:2 132 BEGIN
447 1 10:3 132 REGLINE:=COPY(LINE,1,L);
448 1 10:3 151 WRITELN(REGLINE);
449 1 10:2 171 END;
450 1 10:1 171 L:=L+2;
451 1 10:1 179 NLENGTH:=NLENGTH-L+1;
452 1 10:1 191 IF NLENGTH<1 THEN
453 1 10:2 198 EXIT(SHOWALINE);
454 1 10:1 202 REGLINE:=COPY(LINE,L,NLENGTH);
455 1 10:1 223 FOR I:=1 TO INDENT DO
456 1 10:2 239 WRITE(' ');
457 1 10:1 259 WRITE(REGLINE);
458 1 10:1 271 PC:=PC+1;
459 1 10:0 279 END;
460 1 10:0 298

```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```

461 1 2:0 1 (##P)PROCEDURE BRANCHIN;
462 1 2:0 0 BEGIN
463 1 2:0 0 (##I-#)
464 1 2:1 0 RESET(PASSNODE,'PASSTHRU');
465 1 2:1 18 I:=IORESULT;
466 1 2:1 23 (##I+8)
467 1 2:1 23 IF I<>0 THEN
468 1 2:2 30 BEGIN
469 1 2:3 30 WRITELN('PASSTHRU FILE DOES NOT EXIST');
470 1 2:3 78 WRITELN(' *****FATAL ERROR*****');
471 1 2:3 123 WRITELN(' ',I);
472 1 2:3 167 ANYKEY;
473 1 2:3 169 SETCHAIN('PGM1');
474 1 2:3 179 EXIT(PROGRAM);
475 1 2:2 183 END;
476 1 2:1 183 GET(PASSNODE);
477 1 2:1 190 CURSYS:=PASSNODE^.CURSYS;
478 1 2:1 198 CURSP:=PASSNODE^.CURSP;
479 1 2:1 206 CURSUB:=PASSNODE^.CURSUB;
480 1 2:1 214 PAC:=PASSNODE^.PAC;
481 1 2:1 220 NCURSYS:=PASSNODE^.NCURSYS;
482 1 2:1 227 NCURSP:=PASSNODE^.NCURSP;
483 1 2:1 234 NCURSUB:=PASSNODE^.NCURSUB;
484 1 2:1 241 NPAC:=PASSNODE^.NPAC;
485 1 2:1 248 CLOSE(PASSNODE);
486 1 2:0 256 END;
487 1 2:0 270

```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```

488 1 3:D 1 (86P8)PROCEDURE BRANCHOUT;
489 1 3:0 0 BEGIN
490 1 3:1 0 REWRITE(PASSNODE,'PASSTHRU');
491 1 3:1 20 PASSNODE^.FLAG1:=1;
492 1 3:1 26 PUT(PASSNODE);
493 1 3:1 33 CLOSE(PASSNODE,LOCK);
494 1 3:0 41 END;
495 1 3:0 54
496 1 3:0 54
497 1 3:0 54
498 1 3:0 54 (86I 05:UTILITY.TEXT8)
499 1 3:0 54

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```

500 1 11:D 1 (##P)PROCEDURE CLOSEATTRFILE;
501 1 11:0 0 BEGIN
502 1 11:1 0 RESEY(ATTRFILE,NAMEATCORE);
503 1 11:1 13 FOR I:=1 TO NATTRIBUTES DO
504 1 11:2 29 BEGIN
505 1 11:3 29 ATTRFILE^:=ATTRCORE[I];
506 1 11:3 57 PUT(ATTRFILE);
507 1 11:2 65 END;
508 1 11:1 75 ATTRLAST:=NATTRLAST;
509 1 11:1 92 ATTRFILE^:=ATTRLAST;
510 1 11:1 108 PUT(ATTRFILE);
511 1 11:1 116 CLOSE(ATTRFILE);
512 1 11:0 125 END;
513 1 11:0 140

```

CLOSEATTRFILE saves index to attributes file on the disk.

```

514 1 12:0 1 (80P8)PROCEDURE CLOSEMEASFILE;
515 1 12:0 0 BEGIN
516 1 12:1 0 RESET(MEASFILE,NAMENECORE);
517 1 12:1 13 FOR I:=1 TO NMEASURES DO
518 1 12:2 29 BEGIN
519 1 12:3 29 MEASFILE^:=MEASCORE[I];
520 1 12:3 57 PUT(MEASFILE);
521 1 12:2 65 END;
522 1 12:1 75 MEASLAST:=NMEASLAST;
523 1 12:1 92 MEASFILE^:=MEASLAST;
524 1 12:1 108 PUT(MEASFILE);
525 1 12:1 116 CLOSE(MEASFILE);
526 1 12:0 125 END;
527 1 12:0 140

```

CLOSEMEASFILE saves index to the measures file on the disk.



```

528 1 13:0 1 (#P*)PROCEDURE SETUPSCREEN;
529 1 13:0 0 BEGIN
530 1 13:1 0 I:=TRUNC(CORE[NO] DIV 1000000);
531 1 13:1 54 PAC:=ASPECT[1];
532 1 13:1 72 NPAC:=I;
533 1 13:1 78 TEMP:=CORE[NO] DIV 100;
534 1 13:1 115 TEMP2:=TRUNC(CORE[NO] DIV 10000);
535 1 13:1 151 FOR J:=1 TO NCORELAST DO
536 1 13:2 167 BEGIN
537 1 13:3 167 IF(TEMP2=CORE[J] DIV 10000) AND (CORE[J] DIV 10000*10000=CORE[J]) THEN
538 1 13:4 274 BEGIN
539 1 13:5 274 SEEK(DATANODE,CORE2[J]);
540 1 13:5 298 GET(DATANODE);
541 1 13:5 306 XOBJECTIVE:=DATANODE^.TAXA;
542 1 13:5 316 NOBJECTIVE:=DATANODE^.NTAXA[2];
543 1 13:4 331 END;
544 1 13:3 331 IF(TEMP=CORE[J] DIV 100) AND(CORE[J]DIV 100 * 100=CORE[J]) THEN
545 1 13:4 431 BEGIN
546 1 13:5 431 SEEK(DATANODE,CORE2[J]);
547 1 13:5 455 GET(DATANODE);
548 1 13:5 463 XFUNPUR:=DATANODE^.TAXA;
549 1 13:5 473 NFUNPUR:=DATANODE^.NTAXA[3];
550 1 13:4 488 END;
551 1 13:3 488 IF CORE[NO]=CORE[J] THEN
552 1 13:4 531 BEGIN
553 1 13:5 531 SEEK(DATANODE,CORE2[J]);
554 1 13:5 555 GET(DATANODE);
555 1 13:5 563 XCHARAC:=DATANODE^.TAXA;
556 1 13:5 573 NCHARAC:=DATANODE^.NTAXA[4];
557 1 13:4 588 END;
558 1 13:2 588 END;
559 1 13:0 598 END;
560 1 13:0 614

```

SETUPSCREEN sets up header for the top of each page [or screen] with appropriate information. The header contains the system class, system, subsystem, aspect, objectives, functional purpose and characteristics information for the attributes and/or measures on the display.

```

561 1 14:D 1 (##P#)PROCEDURE TOPPAGE;
562 1 14:0 0 BEGIN
563 1 14:1 0 PAGE(PRNT);
564 1 14:1 10 N:=LENGTH(CURSYS);
565 1 14:1 18 IF M>16 THEN
566 1 14:2 25 M:=16;
567 1 14:1 29 LINE:=COPY(CURSYS,1,M);
568 1 14:1 48 WRITE(PRNT,'*',LINE,' Systems');
569 1 14:1 90 N:=16-LENGTH(CURSYS);
570 1 14:1 100 FOR L:=1 TO N DO
571 1 14:2 116 WRITE(PRNT,' ');
572 1 14:1 136 M:=LENGTH(CURSP);
573 1 14:1 144 IF M>16 THEN
574 1 14:2 151 M:=16;
575 1 14:1 155 LINE:=COPY(CURSP,1,M);
576 1 14:1 174 WRITE(PRNT,'*',LINE);
577 1 14:1 196 N:=16-LENGTH(CURSP);
578 1 14:1 206 FOR L:=1 TO N DO
579 1 14:2 222 WRITE(PRNT,' ');
580 1 14:1 242 M:=LENGTH(CURSUB);
581 1 14:1 250 IF M>16 THEN
582 1 14:2 257 M:=16;
583 1 14:1 261 LINE:=COPY(CURSUB,1,M);
584 1 14:1 280 WRITE(PRNT,'*',LINE);
585 1 14:1 302 N:=16-LENGTH(CURSUB);
586 1 14:1 312 FOR L:=1 TO N DO
587 1 14:2 328 WRITE(PRNT,' ');
588 1 14:1 348 WRITELN(PRNT,'*',FAC);
589 1 14:1 378 IF NPRINT>=1 THEN
590 1 14:2 385 WRITELN(PRNT,'Objective:[',NOBJECTIVE,'J',XOBJECTIVE);
591 1 14:1 450 IF NPRINT>=2 THEN
592 1 14:2 457 WRITELN(PRNT,'Fct1 Prps:[',NFUNPUR,'J',XFUNPUR);
593 1 14:1 522 IF NPRINT=3 THEN
594 1 14:2 529 WRITELN(PRNT,'Charstics:[',NCHARAC,'J',XCHARAC);
595 1 14:1 594 WRITELN(PRNT,' ');
596 1 14:0 612 END;
597 1 14:0 630

```

TOPPAGE prints appropriate header information at the top of each page.

```

598 1 15:0 1  (##P*)PROCEDURE TOPSCREEN;
599 1 15:0 0  BEGIN
600 1 15:1 0  PAGE(OUTPUT);
601 1 15:1 10  M:=LENGTH(CURSYS);
602 1 15:1 18  IF M>16 THEN
603 1 15:2 25  M:=16;
604 1 15:1 29  LINE:=COPY(CURSYS,1,M);
605 1 15:1 48  WRITE('8',LINE,' Systems');
606 1 15:1 90  GOTOXY(26,0);
607 1 15:1 95  M:=LENGTH(CURSP);
608 1 15:1 103 IF M>16 THEN
609 1 15:2 110 M:=16;
610 1 15:1 114 LINE:=COPY(CURSP,1,M);
611 1 15:1 133 WRITE('8',LINE);
612 1 15:1 155 GOTOXY(44,0);
613 1 15:1 160 M:=LENGTH(CURSUR);
614 1 15:1 168 IF M>16 THEN
615 1 15:2 175 M:=16;
616 1 15:1 179 LINE:=COPY(CURSUR,1,M);
617 1 15:1 198 WRITELN('8',LINE);
618 1 15:1 228 GOTOXY(62,0);
619 1 15:1 233 WRITELN('8',PAC);
620 1 15:1 263 M:=LENGTH(XOBJECTIVE);
621 1 15:1 271 IF M>67 THEN M:=67;
622 1 15:1 282 LINE:=COPY(XOBJECTIVE,1,M);
623 1 15:1 301 IF NSCREEN>=1 THEN
624 1 15:2 308 WRITELN('ObjectiveC',NOBJECTIVE,'J:',LINE);
625 1 15:1 376 M:=LENGTH(XFUNPUR);
626 1 15:1 384 IF M>67 THEN M:=67;
627 1 15:1 395 LINE:=COPY(XFUNPUR,1,M);
628 1 15:1 414 IF NSCREEN>=2 THEN
629 1 15:2 421 WRITELN('Fct1 PrpsC',NFUNPUR,'J:',LINE);
630 1 15:1 489 M:=LENGTH(XCHARAC);
631 1 15:1 497 IF M>67 THEN M:=67;
632 1 15:1 508 LINE:=COPY(XCHARAC,1,M);
633 1 15:1 527 IF NSCREEN=3 THEN
634 1 15:2 534 WRITELN('ChrctstcsC',NCHARAC,'J:',LINE);
635 1 15:1 602 WRITELN(' ');
636 1 15:0 620 END;
637 1 15:0 632

```

TOPSCREEN displays appropriate header information at the top of each screen.

```

638 1 16:D 1 (80P8)PROCEDURE PRNTATTRLINE;
639 1 16:0 0 BEGIN
640 1 16:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
641 1 16:1 13 SEEK(ATTRIBUTES,NCURATTRIBUTE);
642 1 16:1 24 GET(ATTRIBUTES);
643 1 16:1 32 K:=ATTRIBUTES^.NDESCRIPTOR;
644 1 16:1 47 LINE:=ATTRIBUTES^.DESCRIPTOR;
645 1 16:1 57 WRITELN(PRNT,LINE,['K,']);
646 1 16:1 109 CLOSE(ATTRIBUTES);
647 1 16:0 118 END;
648 1 16:0 130

```

PRNTATTRLINE prints one attribute when called by PRNTTOP.

```

649 1 17:0 1 (P*)PROCEDURE ATTRLINEDISPLAY;
650 1 17:0 0 BEGIN
651 1 17:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
652 1 17:1 13 SEEK(ATTRIBUTES,NCURATTRIBUTE);
653 1 17:1 24 GET(ATTRIBUTES);
654 1 17:1 32 K:=ATTRIBUTES^.NDESCRIPTOR[5];
655 1 17:1 47 LINE:=ATTRIBUTES^.DESCRIPTOR;
656 1 17:1 57 M:=LENGTH(LINE);
657 1 17:1 65 IF M>67 THEN
658 1 17:2 72 M:=67;
659 1 17:1 76 LINE:=COPY(LINE,1,M);
660 1 17:1 95 WRITELN(LINE,'I',K,'J');
661 1 17:1 147 CLOSE(ATTRIBUTES);
662 1 17:0 156 END;
663 1 17:0 168

```

ATTRLINEDISPLAY adds an attribute to the header for a measurement item display.

```

664 1 18:0 1 (**P)PROCEDURE PRINTONEATTRIBUTE;
665 1 18:0 0 BEGIN
666 1 18:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
667 1 18:1 13 SEEK(ATTRIBUTES,NCURATTRIBUTE);
668 1 18:1 24 GET(ATTRIBUTES);
669 1 18:1 32 WRITE(PRNT,NATTR,'. ');
670 1 18:1 60 FOR J:=1 TO 5 DO
671 1 18:2 74 BEGIN
672 1 18:3 74 K:=ATTRIBUTES^.WDESCRIPTOR[J];
673 1 18:3 91 WRITE(PRNT,K,'. ');
674 1 18:2 113 END;
675 1 18:1 123 LINE:=ATTRIBUTES^.DESCRIPTOR;
676 1 18:1 133 WRITELN(PRNT,'J',LINE);
677 1 18:1 163 CLOSE(ATTRIBUTES);
678 1 18:0 172 END;
679 1 18:0 186

```

PRINTONEATTRIBUTE prints one attribute in the body of the attribute display.

```

680 1 19:D 1 (20P*)PROCEDURE ONEATTRIBUTEDISPLAY;
681 1 19:0 0 BEGIN
682 1 19:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
683 1 19:1 13 SEEK(ATTRIBUTES,NCURATTRIBUTE);
684 1 19:1 24 GET(ATTRIBUTES);
685 1 19:1 32 WRITE(NATTR,' ');
686 1 19:1 60 FOR J:=1 TO 5 DO
687 1 19:2 74 BEGIN
688 1 19:3 74 K:=ATTRIBUTES^.NDESCRIPTOR(J);
689 1 19:3 91 WRITE(K,' ');
690 1 19:2 113 END;
691 1 19:1 123 LINE:=ATTRIBUTES^.DESCRIPTOR;
692 1 19:1 133 LLENGTH:=60;
693 1 19:1 137 WRITE('J');
694 1 19:1 147 INDENT:=16;
695 1 19:1 151 SHOWALINE;
696 1 19:1 153 WRITELN(' ');
697 1 19:1 171 CLOSE(ATTRIBUTES);
698 1 19:0 180 END;
699 1 19:0 194

```

ONEATTRIBUTEDISPLAY displays one attribute in the body of the attribute display.

```

700 1 20:0 1 (80P)PROCEDURE PRINTTHEATTRIBUTES;
701 1 20:0 0 BEGIN
702 1 20:1 0 NATTR:=0;
703 1 20:1 4 OK:=FALSE;
704 1 20:1 8 WRITELN(PRNT,'Measurable Attributes--To evaluate effectiveness in meeting this ',
705 1 20:1 85 chr(13),' characteristic, the following system attributes can be
measured:');
706 1 20:1 182 FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
707 1 20:2 198 BEGIN
708 1 20:3 198 TEMPX:=ATTRCORE[NCURATTRIBUTE] DIV 100;
709 1 20:3 235 TEMP:=TEMPX;
710 1 20:3 251 IF TEMP=CORE[NCURATTRIBUTE] THEN
711 1 20:4 282 BEGIN
712 1 20:5 282 OK:=TRUE;
713 1 20:5 286 NATTR:=NATTR+1;
714 1 20:5 294 PRINTONEATTRIBUTE;
715 1 20:4 296 END;
716 1 20:2 296 END;
717 1 20:1 306 IF OK=FALSE THEN
718 1 20:2 314 WRITELN(PRNT,' ...none');
719 1 20:0 344 END;
720 1 20:0 358

```

PRINTTHEATTRIBUTES prints the body of the attribute display on the printer.



```

721 1 21:0 1 (88Ps)PROCEDURE SHOWATTRIBUTES;
722 1 21:0 0 BEGIN
723 1 21:1 0 NATTR:=0;
724 1 21:1 4 FOR I:=1 TO 20 DO
725 1 21:2 18 BEGIN
726 1 21:3 18 ATTRINDEX[I]:=0;
727 1 21:3 33 MEASINDEX[I]:=0;
728 1 21:2 48 END;
729 1 21:1 58 GOTOXY(0,4);
730 1 21:1 63 WRITE(CHR(11));
731 1 21:1 73 OK:=FALSE;
732 1 21:1 77 DISPCOUNT:=0;
733 1 21:1 81 WRITELN('Measurable Attributes--To evaluate effectiveness in meeting this ',
734 1 21:1 158 chr(13),' characteristic, the following attributes can be measured:');
735 1 21:1 255 FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
736 1 21:2 271 BEGIN
737 1 21:3 271 TEMPX:=ATTRCORE[NCURATTRIBUTE] DIV 100;
738 1 21:3 308 TEMP:=TEMPX;
739 1 21:3 324 IF TEMP=CORE[MODE] THEN
740 1 21:4 355 BEGIN
741 1 21:5 355 IF DISPCOUNT >=10 THEN
742 1 21:6 362 BEGIN
743 1 21:7 362 DISPCOUNT:=0;
744 1 21:7 366 ANYKEY;
745 1 21:7 368 GOTOXY(0,6);
746 1 21:7 373 WRITE(CHR(11));
747 1 21:6 383 END;
748 1 21:5 383 OK:=TRUE;
749 1 21:5 387 NATTR:=NATTR+1;
750 1 21:5 395 ATTRINDEX[NATTR]:=NCURATTRIBUTE;
751 1 21:5 412 PC:=0;
752 1 21:5 416 ONEATTRIBUTEDISPLAY;
753 1 21:5 418 IF PC=1 THEN
754 1 21:6 425 DISPCOUNT:=DISPCOUNT+1;
755 1 21:5 433 PC:=0;
756 1 21:5 437 DISPCOUNT:=DISPCOUNT+1;
757 1 21:4 445 END;
758 1 21:2 445 END;
759 1 21:1 455 IF OK=FALSE THEN
760 1 21:2 463 WRITELN(' ...none');
761 1 21:0 493 END;
762 1 21:0 512

```

SHOWATTRIBUTES displays the body of the attribute display on the screen.

```

763 1 22:0 1 (**P*)PROCEDURE REWORDATTRIBUTES;
764 1 22:0 0 BEGIN
765 1 22:1 0 IF OK THEN
766 1 22:2 5 BEGIN
767 1 22:3 5 REPEAT
768 1 22:4 5 GOTOXY(0,15);
769 1 22:4 10 WRITE(CHR(11));
770 1 22:4 20 WRITE('Which one (type 0 if done) ?');
771 1 22:4 40 KEYN;
772 1 22:4 42 IF (I<0) OR (I>NATTR) THEN
773 1 22:5 77 BEGIN
774 1 22:6 77 WRITELN('Please type an integer between 0 and ',NATTR,'. ');
775 1 22:6 156 ANYKEY;
776 1 22:5 158 END;
777 1 22:3 158 UNTIL (I>=0) AND (I<=NATTR);
778 1 22:3 173 IF I<>0 THEN
779 1 22:4 180 BEGIN
780 1 22:5 180 I:=ATTRINDEX[I];
781 1 22:5 197 WRITELN('Please type the new attribute descriptor: ');
782 1 22:5 259 WRITE('.....');
783 1 22:5 284 INLINE;
784 1 22:5 286 RESET(ATTRIBUTES,NAMEATTRIBUTES);
785 1 22:5 299 SEEK(ATTRIBUTES,I);
786 1 22:5 310 GET(ATTRIBUTES);
787 1 22:5 318 IF LENGTH(LINER)<69 THEN
788 1 22:6 327 ATTRIBUTES^.DESCRIPTOR:=LINER
789 1 22:5 332 ELSE
790 1 22:6 339 ATTRIBUTES^.DESCRIPTOR:=COPY(LINER,1,68);
791 1 22:5 358 SEEK(ATTRIBUTES,I);
792 1 22:5 369 PUT(ATTRIBUTES);
793 1 22:5 377 CLOSE(ATTRIBUTES);
794 1 22:4 386 END;
795 1 22:2 386 END
796 1 22:1 386 ELSE
797 1 22:2 388 BEGIN
798 1 22:3 388 WRITELN('There are no attributes for this performance item');
799 1 22:3 457 ANYKEY;
800 1 22:2 459 END;
801 1 22:0 459 END;
802 1 22:0 478
803 1 22:0 478

```

REWORDATTRIBUTES asks which attribute to reword. Then it asks the analyst to reword the attribute.

```

004 1 23:0 1 (*6*)PROCEDURE DELETEATTRIBUTES;
005 1 23:0 0 BEGIN
006 1 23:1 0 IF OK THEN
007 1 23:2 5 BEGIN
008 1 23:3 5 REPEAT
009 1 23:4 5 GOTOXY(0,15);
010 1 23:4 10 WRITE(CMR(11));
011 1 23:4 20 WRITE('Which one (type 0 if done) ?');
012 1 23:4 60 KEYN;
013 1 23:4 62 IF (I<0) OR (I>NATTR) THEN
014 1 23:5 77 BEGIN
015 1 23:6 77 WRITELN('Please type an integer between 0 and ',NATTR,',');
016 1 23:6 156 ANYKEY;
017 1 23:5 158 END;
018 1 23:3 158 UNTIL (I>=0) AND (I<=NATTR);
019 1 23:3 173 IF I<>0 THEN
020 1 23:4 180 BEGIN
021 1 23:5 180 I:=ATTRINDEX(I);
022 1 23:5 197 RESET(ATTRIBUTES,NAMEATTRIBUTES);
023 1 23:5 210 SEEK(ATTRIBUTES,I);
024 1 23:5 221 GET(ATTRIBUTES);
025 1 23:5 229 FOR J:=1 TO 6 DO
026 1 23:6 243 ATTRIBUTES^.NDESCRIPTOR(J):=0;
027 1 23:5 268 ATTRIBUTES^.DESCRIPTOR:=
028 1 23:5 298 SEEK(ATTRIBUTES,I);
029 1 23:5 309 PUT(ATTRIBUTES);
030 1 23:5 317 CLOSE(ATTRIBUTES);
031 1 23:5 326 ATTRCOREC(I):=0;
032 1 23:4 353 END;
033 1 23:2 353 END
034 1 23:1 353 ELSE
035 1 23:2 355 BEGIN
036 1 23:3 355 WRITELN('There are no attributes for this performance item');
037 1 23:3 424 ANYKEY;
038 1 23:2 426 END;
039 1 23:0 426 END;
040 1 23:0 446

```

DELETEATTRIBUTES asks analyst which attribute to delete. Then it deletes the attribute.

```

041 1 24:0 1 (88P8)PROCEDURE ADDATTRIBUTES;
042 1 24:0 0 BEGIN
043 1 24:1 0 IF MATRLAST>=200 THEN
044 1 24:2 9 BEGIN
045 1 24:3 9 WRITELN('DATASET CONTAINS 200 ATTRIBUTE LIMIT');
046 1 24:3 65 ANYKEY;
047 1 24:2 67 END;
048 1 24:1 67 FOR J:=1 TO 20 DO
049 1 24:2 81 SCRATCH[J]:=J;
050 1 24:1 108 FOR J:=1 TO MATRLAST DO
051 1 24:2 124 IF CORE[MODE]=(ATTRCORE[J] DIV 100) THEN
052 1 24:3 176 BEGIN
053 1 24:4 176 K:=TRUNC(ATTRCORE[J]-ATTRCORE[J] DIV 100 * 100);
054 1 24:4 241 IF K<>0 THEN
055 1 24:5 248 SCRATCH[K]:=0;
056 1 24:3 263 END;
057 1 24:3 273 (88I-8)
058 1 24:1 273 RESET(ATTRIBUTES,NAMEATTRIBUTES);
059 1 24:1 284 (88I+8)
060 1 24:1 284 FOR J:=1 TO 4 DO
061 1 24:2 298 ATTRIBUTES^.NDESCRIPTOR[J]:=DATANODE^.NTAXA[J];
062 1 24:1 336 ATTRIBUTES^.NDESCRIPTOR[6]:=0;
063 1 24:1 349 GOTOXY(0,15);
064 1 24:1 354 WRITE(CHR(11));
065 1 24:1 364 WRITELN('Please type the new attribute descriptor:');
066 1 24:1 425 WRITE('.....');
067 1 24:1 450 INLINE;
068 1 24:1 452 IF LINER='' THEN
069 1 24:2 462 BEGIN
070 1 24:3 462 CLOSE(ATTRIBUTES);
071 1 24:3 471 EXIT(ADDATTRIBUTES);
072 1 24:2 475 END;
073 1 24:1 475 MATRLAST:=MATRLAST+1;
074 1 24:1 483 SEEK(ATTRIBUTES,MATRLAST);
075 1 24:1 494 FOR J:=20 DOWNT0 1 DO
076 1 24:2 508 IF SCRATCH[J]<>0 THEN
077 1 24:3 526 BEGIN
078 1 24:4 526 ATTRIBUTES^.NDESCRIPTOR[5]:=J;
079 1 24:4 541 K:=J;
080 1 24:3 547 END;

```

ADDATTRIBUTES asks the analyst to type in a new attribute, then it adds the attribute to the attribute list.

```

081 1 24:1 557 IF LENGTH(LINER)<69 THEN
082 1 24:2 566 ATTRIBUTES^.DESCRIPTOR:=LINER
083 1 24:1 571 ELSE
084 1 24:2 578 ATTRIBUTES^.DESCRIPTOR:=COPY(LINER,1,68);
085 1 24:1 597 PUT(ATTRIBUTES);
086 1 24:1 605 TEMPX:=CORE[NOBJ]*100+K;
087 1 24:1 653 SCRATCHKJ:=0;
088 1 24:1 668 ATTRCOREINATTRLASTJ:=TEMPX;
089 1 24:1 696 CLOSE(ATTRIBUTES);
090 1 24:1 705 OK:=TRUE;
091 1 24:0 709 END;
092 1 24:0 732
092 1 24:0 732 (*01 05:MEASATTR2.TEXT *)

```

See previous page for program description.

```

893 1 25:0 1 (89P)PROCEDURE PRINTAMEASURE;
894 1 25:0 0 BEGIN
895 1 25:1 0 RESET(MEASURES,NAME MEASURES);
896 1 25:1 13 SEEK(MEASURES,NCURMEASURE);
897 1 25:1 24 GET(MEASURES);
898 1 25:1 32 WRITE(PRINT,NMEAS,'. ');
899 1 25:1 60 FOR J:=1 TO '6 DO
900 1 25:2 74 BEGIN
901 1 25:3 74 K:=MEASURES^.NDESCRIPTOR(J);
902 1 25:3 91 WRITE(PRINT,K,'. ');
903 1 25:2 113 END;
904 1 25:1 123 LINE:=MEASURES^.DESCRIPTOR;
905 1 25:1 133 WRITELN(PRINT,'J',LINE);
906 1 25:1 163 CLOSE(MEASURES);
907 1 25:0 172 END;
908 1 25:0 186

```

PRINTAMEASURE prints one measure in the body of the measure display.

```

909 1 26:D 1 (80P8)PROCEDURE ONEMEASUREDISPLAY;
910 1 26:0 0 BEGIN
911 1 26:1 0 RESET(MEASURES,NAMEMEASURES);
912 1 26:1 13 SEEK(MEASURES,NCURMEASURE);
913 1 26:1 24 GET(MEASURES);
914 1 26:1 32 WRITE(NMEAS,'. ');
915 1 26:1 60 FOR J:=1 TO 6 DO
916 1 26:2 74 BEGIN
917 1 26:3 74 K:=MEASURES^.NDESCRIPTOR[J];
918 1 26:3 91 WRITE(K,'. ');
919 1 26:2 113 END;
920 1 26:1 123 LINE:=MEASURES^.DESCRIPTOR;
921 1 26:1 133 LLENGTH:=60;
922 1 26:1 137 WRITE(' ');
923 1 26:1 147 INDENT:=18;
924 1 26:1 151 SHOWALINE;
925 1 26:1 153 WRITELN(' ');
926 1 26:1 171 CLOSE(MEASURES);
927 1 26:0 180 END;
928 1 26:0 194

```

ONEMEASUREDISPLAY displays one measure in the body of the measure display.

```

929 1 27:0 1 (#P*)PROCEDURE PRNTTHEMEASURES;
930 1 27:0 0 BEGIN
931 1 27:1 0 NMEAS:=0;
932 1 27:1 4 OK:=FALSE;
933 1 27:1 8 WRITELN(PRNT,'Measures--This system attribute can be analyzed by comparing',
934 1 27:1 80 chr(13),' the following parameters with established criteria:');
935 1 27:1 163 FOR NCURMEASURE:=1 TO NMEASLAST DO
936 1 27:2 179 BEGIN
937 1 27:3 179 TEMPX:=MEASCORE[NCURMEASURE] DIV 100;
938 1 27:3 216 IF TEMPX=ATTRCORE[NCURATTRIBUTE] THEN
939 1 27:4 247 BEGIN
940 1 27:5 247 OK:=TRUE;
941 1 27:5 251 NMEAS:=NMEAS+1;
942 1 27:5 259 PRINTAMEASURE;
943 1 27:4 261 END;
944 1 27:2 261 END;
945 1 27:1 271 IF OK=FALSE THEN
946 1 27:2 279 WRITELN(PRNT,' ...none');
947 1 27:0 309 END;
948 1 27:0 324

```

PRNTTHEMEASURES prints the body of the measure display.



```

949 1 28:0 1 (80P*)PROCEDURE SHOWMEASURES;
950 1 28:0 0 BEGIN
951 1 28:1 0 NMEAS:=0;
952 1 28:1 4 FOR I:=1 TO 20 DO
953 1 28:2 18 MEASINDEX[I]:=0;
954 1 28:1 43 GOTOXY(0,7);
955 1 28:1 48 WRITE(CHR(13));
956 1 28:1 58 OK:=FALSE;
957 1 28:1 62 DISPMCOUNT:=1;
958 1 28:1 66 WRITELN('Measures--This system attribute can be analyzed by comparing',
959 1 28:1 138 chr(13),' the following parameters with established criteria:');
960 1 28:1 221 FOR NCURMEASURE:=1 TO NMEASLAST DO
961 1 28:2 237 BEGIN
962 1 28:3 237 TEMPX:=MEASCORE[NCURMEASURE] DIV 100;
963 1 28:3 274 IF TEMPX=ATTRCORE[NCURATTRIBUTE] THEN
964 1 28:4 305 BEGIN
965 1 28:5 305 IF DISPMCOUNT > 6 THEN
966 1 28:6 312 BEGIN
967 1 28:7 312 DISPMCOUNT:=1;
968 1 28:7 316 ANYKEY;
969 1 28:7 318 GOTOXY(0,9);
970 1 28:7 323 WRITE(CHR(11));
971 1 28:6 333 END;
972 1 28:5 333 OK:=TRUE;
973 1 28:5 337 NMEAS:=NMEAS+1;
974 1 28:5 345 MEASINDEX[NMEAS]:=NCURMEASURE;
975 1 28:5 362 PC:=0;
976 1 28:5 366 ONEMEASUREDISPLAY;
977 1 28:5 368 IF PC=1 THEN
978 1 28:6 375 DISPMCOUNT:=DISPMCOUNT+1;
979 1 28:5 383 PC:=0;
980 1 28:5 387 DISPMCOUNT:=DISPMCOUNT+1;
981 1 28:4 395 END;
982 1 28:2 395 END;
983 1 28:1 405 IF OK=FALSE THEN
984 1 28:2 413 WRITELN(' ...none');
985 1 28:0 443 END;
986 1 28:0 462

```

SHOWMEASURES displays the body of the measure display on the screen.

```

987 1 29:0 1 (80P)PROCEDURE REWORDMEASURES;
988 1 29:0 0 BEGIN
989 1 29:1 0 REPEAT
990 1 29:2 0 GOTOXY(0,15);
991 1 29:2 5 WRITE(CHR(11));
992 1 29:2 15 WRITE('Which one (type 0 if done) ?');
993 1 29:2 55 KEYN;
994 1 29:2 57 IF (I<0) OR (I>NMEAS) THEN
995 1 29:3 72 BEGIN
996 1 29:4 72 WRITELN('please type an integer between 0 and ',NMEAS,',');
997 1 29:4 151 ANYKEY;
998 1 29:3 153 END;
999 1 29:1 153 UNTIL (I>=0) AND (I<=NMEAS);
1000 1 29:1 168 IF I<>0 THEN
1001 1 29:2 175 BEGIN
1002 1 29:3 175 I:=MEASINDEX[I];
1003 1 29:3 192 WRITELN('Please type the new measure descriptor: ');
1004 1 29:3 252 WRITE('.....');
1005 1 29:3 277 INLINE;
1006 1 29:3 279 RESET(MEASURES,NAMEMEASURES);
1007 1 29:3 292 SEEK(MEASURES,I);
1008 1 29:3 303 GET(MEASURES);
1009 1 29:3 311 IF LENGTH(LINER)<69 THEN
1010 1 29:4 320 MEASURES^.DESCRIPTOR:=LINER
1011 1 29:3 325 ELSE
1012 1 29:4 332 MEASURES^.DESCRIPTOR:=COPY(LINER,1,68);
1013 1 29:3 351 SEEK(MEASURES,I);
1014 1 29:3 362 PUT(MEASURES);
1015 1 29:3 370 CLOSE(MEASURES);
1016 1 29:2 379 END
1017 1 29:1 379 ELSE
1018 1 29:2 381 BEGIN
1019 1 29:3 381 WRITELN('There are no measures for this attribute');
1020 1 29:3 441 ANYKEY;
1021 1 29:2 443 END;
1022 1 29:0 443 END;
1023 1 29:0 460
1024 1 29:0 460

```

REWORDMEASURES asks analyst which measure to reword. Then it asks him/her to reword the measure.

```

1025 1 30:D 1 (##P*)PROCEDURE DELETEMEASURES;
1026 1 30:0 0 BEGIN
1027 1 30:1 0 REPEAT
1028 1 30:2 0 GOTOXY(0,15);
1029 1 30:2 5 WRITE(CHR(11));
1030 1 30:2 15 WRITE('Which one (type 0 if done) ?');
1031 1 30:2 55 KEYN;
1032 1 30:2 57 IF (I<0) OR (I>NMEAS) THEN
1033 1 30:3 72 BEGIN
1034 1 30:4 72 WRITELN('Please type an integer between 0 and ',NMEAS,'. ');
1035 1 30:4 151 ANYKEY;
1036 1 30:3 153 END;
1037 1 30:1 153 UNTIL (I>=0) OR (I<=NMEAS);
1038 1 30:1 168 IF I<>0 THEN
1039 1 30:2 175 BEGIN
1040 1 30:3 175 I:=MEASINDEX[I];
1041 1 30:3 192 RESET(MEASURES,NAMEMEASURES);
1042 1 30:3 205 SEEK(MEASURES,I);
1043 1 30:3 216 GET(MEASURES);
1044 1 30:3 224 FOR J:=1 TO 6 DO
1045 1 30:4 238 MEASURES^.NDESCRIPTOR[J]:=0;
1046 1 30:3 263 MEASURES^.DESCRIPTOR:=
1047 1 30:3 293 SEEK(MEASURES,I);
1048 1 30:3 304 PUT(MEASURES);
1049 1 30:3 312 CLOSE(MEASURES);
1050 1 30:3 321 MEASCORE[I]:=0;
1051 1 30:2 348 END
1052 1 30:1 348 ELSE
1053 1 30:2 350 BEGIN
1054 1 30:3 350 WRITELN('There are no measures for this attribute');
1055 1 30:3 410 ANYKEY;
1056 1 30:2 412 END;
1057 1 30:0 412 END;
1058 1 30:0 430

```

DELETEMEASURES asks the analyst which measure to delete. Then it deletes it.

```

1059 1 31:D 1 (SP)PROCEDURE ADDMEASURES;
1060 1 31:0 0 BEGIN
1061 1 31:1 0 IF NMEASLAST>=400 THEN
1062 1 31:2 9 BEGIN
1063 1 31:3 9 Writeln('DATA SET CONTAINS LIMIT OF 400 MEASURES');
1064 1 31:3 48 ANYKEY;
1065 1 31:3 70 EXIT(ADDMEASURES);
1066 1 31:2 74 END;
1067 1 31:1 74 FOR J:=1 TO 20 DO
1068 1 31:2 88 SCRATCHEJ:=J;
1069 1 31:1 115 FOR J:=1 TO NMEASLAST DO
1070 1 31:2 131 IF ATTRCORE[NCURATTRIBUTE]=(MEASCORE[J] DIV 100) THEN
1071 1 31:3 183 BEGIN
1072 1 31:4 183 K:=TRUNC(MEASCORE[J]-MEASCORE[J] DIV 100 * 100);
1073 1 31:4 248 IF K<>0 THEN
1074 1 31:5 255 SCRATCHCK:=0;
1075 1 31:3 270 END;
1076 1 31:3 280 (SP)-8)
1077 1 31:1 280 RESET(MEASURES,NAMEMEASURES);
1078 1 31:1 291 (SP)+8)
1079 1 31:1 291 FOR J:=1 TO 5 DO
1080 1 31:2 305 MEASURES^.NDESCRIPTOR[J]:=ATTRIBUTES^.NDESCRIPTOR[J];
1081 1 31:1 343 GOTOXY(0,15);
1082 1 31:1 348 WRITE(CHR(11));
1083 1 31:1 358 Writeln('Please type the new measure descriptor (68 characters available:');
1084 1 31:1 442 WRITE('.....');
1085 1 31:1 467 INLINE;
1086 1 31:1 469 IF LINER='' THEN
1087 1 31:2 479 BEGIN
1088 1 31:3 479 CLOSE(MEASURES);
1089 1 31:3 488 EXIT(ADDMEASURES);
1090 1 31:2 492 END;
1091 1 31:1 492 NMEASLAST:=NMEASLAST+1;
1092 1 31:1 500 SEEK(MEASURES,NMEASLAST);
1093 1 31:1 511 FOR J:=20 DOWNT0 1 DO
1094 1 31:2 525 IF SCRATCHEJ<>0 THEN
1095 1 31:3 543 BEGIN
1096 1 31:4 543 MEASURES^.NDESCRIPTOR[6]:=J;
1097 1 31:4 558 K:=J;
1098 1 31:3 564 END;

```

ADDMEASURES asks analyst to type in the new measures.

```

1099 1 31:1 574 if length(liner)<69 then
1100 1 31:2 583 MEASURES^.DESCRIPTOR:=LINER
1101 1 31:1 588 else
1102 1 31:2 595 MEASURES^.descriptor:=copy(liner,1,68);
1103 1 31:1 614 PUT(MEASURES);
1104 1 31:1 622 TEMPX:=ATTRCORE[NCURATTRIBUTE]*100+K;
1105 1 31:1 670 SCRATCH[K]:=0;
1106 1 31:1 685 MEASCORE[MEASLAST]:=TEMPX;
1107 1 31:1 713 CLOSE(MEASURES);
1108 1 31:0 722 END;
1109 1 31:0 744

```

See previous page for program description.

```

1110 1 32:D 1 (80P)PROCEDURE PRINTMEASURES;
1111 1 32:0 0 BEGIN
1112 1 32:1 0 REWRITE(PRNT,'PRINTER:');
1113 1 32:1 21 TOPPAGE;
1114 1 32:1 23 WRITELN(PRNT,'Measurable Attributes--To evaluate effectiveness in meeting this',
1115 1 32:1 99 chr(13),' characteristic, the following system attributes can be
measured: ');
1116 1 32:1 196 WRITE(' ');
1117 1 32:1 212 PRNTATTALINE;
1118 1 32:1 214 PRNTTHEMEASURES;
1119 1 32:1 216 CLOSE(PRNT);
1120 1 32:0 225 END;
1121 1 32:0 238

```

PRINTMEASURES controls printout of entire page of measures.

```

1122 1 33:0 1 (*0P8)PROCEDURE EXAMINEMEASURES;
1123 1 33:0 0 BEGIN
1124 1 33:1 0 TOPSCREEN;
1125 1 33:1 2 GOTOXY(0,4);
1126 1 33:1 7 WRITE(CHR(11));
1127 1 33:1 17 WRITELN('Measurable Attributes--To evaluate effectiveness in meeting this',
1128 1 33:1 93 chr(13),' characteristic, the following system attributes can be measured: ');
1129 1 33:1 190 WRITE(' ');
1130 1 33:1 206 ATTRLINEDISPLAY;
1131 1 33:1 208 SHOWMEASURES;
1132 1 33:1 210 GOTOXY(0,15);
1133 1 33:1 215 WRITE(CHR(11));
1134 1 33:1 225 WRITE('You may perform the following procedures:',chr(13),
1135 1 33:1 288 ' 1. Add new measures 2. Reward a measure',chr(13),
1136 1 33:1 363 ' 3. Remove a measure 4. Print these measures',chr(13),
1137 1 33:1 442 ' 5. Return to Attributes Level ',chr(13),
1138 1 33:1 498 'Please select one: ');
1139 1 33:1 529 REPEAT
1140 1 33:2 529 KEYN;
1141 1 33:2 531 IF (I<1) OR (I>5) THEN
1142 1 33:3 544 WRITELN('Please type an integer between 1 and 5');
1143 1 33:1 602 UNTIL (I>=1) AND (I<=5);
1144 1 33:1 615 CASE 1 OF
1145 1 33:1 620 1:ADDMEASURES;
1146 1 33:1 624 2:REWARDMEASURES;
1147 1 33:1 628 3:DELETEMEASURES;
1148 1 33:1 632 4:PRINTMEASURES;
1149 1 33:1 636 5:EXIT(EXAMINEMEASURES);
1150 1 33:1 642 END;
1151 1 33:1 660 EXAMINEMEASURES;
1152 1 33:0 662 END;
1153 1 33:0 676
1154 1 33:0 676 (*01 05:MEASATTR2.TEXT *)
1155 1 33:0 676
1155 1 33:0 676 (*01 05:MEASATTR3.TEXT *)

```

EXAMINEMEASURES controls production of entire display for analyzing measures.

```

1154 1 34:D 1 (89P8)PROCEDURE ONEPERFITEMDISPLAY;
1157 1 34:0 0 BEGIN
1158 1 34:1 0 SEEK(DATANODE,CORE2[MODE]);
1159 1 34:1 24 GET(DATANODE);
1160 1 34:1 32 K:=DATANODE^.NTAXA[CH];
1161 1 34:1 49 LINE:=DATANODE^.TAXA;
1162 1 34:1 59 LLENGTH:=72;
1163 1 34:1 63 IF K<>0 THEN
1164 1 34:2 70 BEGIN
1165 1 34:3 70 WRITE(' ',K,' ');
1166 1 34:3 110 INDENT:=6;
1167 1 34:3 114 SHOWALINE;
1168 1 34:3 116 WRITELN(' ');
1169 1 34:2 134 END;
1170 1 34:1 134 IF (K=0) AND (M=3) THEN
1171 1 34:2 147 WRITELN(' ',K,' ','Process at the Objectives level');
1172 1 34:1 238 IF (K=0) AND (M=4) THEN
1173 1 34:2 251 WRITELN(' ',K,' ','Process at the Functional Purposes level');
1174 1 34:0 351 END;
1175 1 34:0 364

```

ONEPERFITEMDISPLAY displays one performance item in the body of the display used to select which performance item ought to be processed next.



```

1176 1 35:D 1 (##P)PROCEDURE SHOWPERFITEMS;
1177 1 35:0 0 BEGIN
1178 1 35:1 0 OK:=FALSE;
1179 1 35:1 4 DISPCOUNT:=0;
1180 1 35:1 8 IF M=2 THEN TEMPL1:=1000000;
1181 1 35:1 50 IF M=2 THEN TEMPL3:=10000;
1182 1 35:1 74 IF M=3 THEN TEMPL1:=10000;
1183 1 35:1 98 IF M=3 THEN TEMPL3:=100;
1184 1 35:1 120 IF M=4 THEN TEMPL1:=100;
1185 1 35:1 142 IF M=4 THEN TEMPL3:=1;
1186 1 35:1 164 TEMPL2:=TEMP DIV TEMPL1;
1187 1 35:1 190 FOR NODE:=1 TO NCORELAST DO
1188 1 35:2 206 BEGIN
1189 1 35:3 206 IF (TEMPL2=CORE[NODE] DIV TEMPL1) AND
1190 1 35:3 245 (CORE[NODE] DIV TEMPL3 * TEMPL3 = CORE[NODE]) THEN
1191 1 35:4 309 BEGIN
1192 1 35:5 309 OK:=TRUE;
1193 1 35:5 313 ONEPERFITEMDISPLAY;
1194 1 35:5 315 DISPCOUNT:=DISPCOUNT+1;
1195 1 35:4 323 END;
1196 1 35:3 323 IF (DISPCOUNT DIV 10 * 10=DISPCOUNT) AND (DISPCOUNT<>0) THEN
1197 1 35:4 342 BEGIN
1198 1 35:5 342 DISPCOUNT:=0;
1199 1 35:5 346 ANYKEY;
1200 1 35:5 348 GOTOXY(0,5);
1201 1 35:5 353 WRITE(CHR(11));
1202 1 35:4 363 END;
1203 1 35:2 363 END;
1204 1 35:1 373 IF OK=FALSE THEN
1205 1 35:2 381 WRITELN(' ...none');
1206 1 35:0 411 END;
1207 1 35:0 428

```

SHOWPERFITEMS controls production of the body of displays of performance items.

```

1208 136:D 1 (80P)PROCEDURE CHANGECHARACTERISTICS;
1209 136:0 0 BEGIN
1210 136:1 0 TOPSCREEN;
1211 136:1 2 GOTOXY(0,4);
1212 136:1 7 WRITE(CMR(1));
1213 136:1 17 M:=4;
1214 136:1 21 WRITELN('The following Characteristics are available for the Functional Purpose');
1215 136:1 111 SHOWPERFITENS;
1216 136:1 113 WRITE('Please select one: ');
1217 136:1 144 KEYN;
1218 136:1 146 MCHARAC:=1;
1219 136:1 152 TEMPL4:=1;
1220 136:1 169 TEMP:=TEMP+TEMPL4;
1221 136:1 195 FOR I:=1 TO 300 DO
1222 136:2 211 IF TEMP=CORE[I] THEN
1223 136:3 242 BEGIN
1224 136:4 242 SEEK(DATANODE,CORE2[I]);
1225 136:4 266 GET(DATANODE);
1226 136:4 274 XCHARAC:=DATANODE^.TAXA;
1227 136:3 284 END;
1228 136:1 294 NSCREEN:=3;
1229 136:1 298 NPRINT:=3;
1230 136:0 302 END;
1231 136:0 316

```

CHANGECHARACTERISTICS governs producing the list of characteristics when analyst is selecting a different performance item.

```

1232 1 37:D  X**P*)PROCEDURE CHANGEFUNCTIONALPURPOSES;
1233 1 37:0  0 BEGIN
1234 1 37:1  0 TOPSCREEN;
1235 1 37:1  2 GOTOXY(0,4);
1236 1 37:1  7 WRITE(CHR(11));
1237 1 37:1  17 N:=3;
1238 1 37:1  21 WRITELN('The following Functional Purposes are available for the objective selected: ');
1239 1 37:1  117 SHOWPERFITEMS;
1240 1 37:1  119 WRITE('Please select one: ');
1241 1 37:1  150 KEYN;
1242 1 37:1  152 TEMPL4:=I;
1243 1 37:1  169 NFUNPUR:=I;
1244 1 37:1  175 TEMP:=TEMP+TEMPL4*100;
1245 1 37:1  210 FOR I:=1 TO 300 DO
1246 1 37:2  226 IF TEMP=CORE[I] THEN
1247 1 37:3  257 BEGIN
1248 1 37:4  257 SEEK(DATANODE,CORE2[I]);
1249 1 37:4  281 GET(DATANODE);
1250 1 37:4  289 XFUNPUR:=DATANODE^.TAXA;
1251 1 37:3  299 END;
1252 1 37:1  309 IF NFUNPUR<>0 THEN
1253 1 37:2  316 BEGIN
1254 1 37:3  316 NSCREEN:=2;
1255 1 37:3  320 NPRINT:=2;
1256 1 37:3  324 CHANGECHARACTERISTICS;
1257 1 37:2  326 END;
1258 1 37:0  326 END;
1259 1 37:0  340

```

CHANGEFUNCTIONALPURPOSES governs producing a list of functional purposes when analyst is selecting a different performance item.

```

1260 1 38:D 1 (##P*)PROCEDURE CHANGENODE;
1261 1 38:0 0 BEGIN
1262 1 38:1 0 REPEAT
1263 1 38:2 0 NSCREEN:=0;
1264 1 38:2 4 NPRINT:=0;
1265 1 38:2 8 PAC:= ' ';
1266 1 38:2 28 TOPSCREEN;
1267 1 38:2 30 GOTOXY(0,4);
1268 1 38:2 35 WRITE(CHR(11));
1269 1 38:2 45 WRITE('The following aspects are part of the APH:',chr(13),
1270 1 38:2 109 ' 1. Potentialities',chr(13),
1271 1 38:2 150 ' 2. Processes',chr(13),
1272 1 38:2 184 ' 3. Products',chr(13),
1273 1 38:2 221 ' 4. Environment',chr(13),
1274 1 38:2 259 ' 5. Constraints',chr(13),
1275 1 38:2 297 'Please select one: ');
1276 1 38:2 328 KEYN;
1277 1 38:2 330 PAC:=ASPECT[I];
1278 1 38:2 348 NPAC:=I;
1279 1 38:2 354 TEMPL4:=I;
1280 1 38:2 371 TEMP:=TEMPL4*1000000;
1281 1 38:2 416 GOTOXY(0,0);
1282 1 38:2 421 WRITE(CHR(11));
1283 1 38:2 431 TOPSCREEN;
1284 1 38:2 433 N:=2;
1285 1 38:2 437 WRITELN('The following Objectives are available for the aspect selected: ');
1286 1 38:2 521 SHOWPERFITEMS;
1287 1 38:2 523 WRITE('Please select one: ');
1288 1 38:2 554 KEYN;
1289 1 38:2 556 NOBJECTIVE:=I;
1290 1 38:2 562 TEMPL4:=I;
1291 1 38:2 579 TEMP:=TEMP+TEMPL4*10000;
1292 1 38:2 616 FOR I:=1 TO 300 DO
1293 1 38:3 632 IF TEMP=CORE[I] THEN
1294 1 38:4 663 BEGIN
1295 1 38:5 663 SEEK(DATANODE,CORE2[I]);
1296 1 38:5 687 GET(DATANODE);
1297 1 38:5 695 XORJECTIVE:=DATANODE^.TAXA;
1298 1 38:4 705 END;
1299 1 38:2 715 IF NOBJECTIVE<>0 THEN

```

CHANGENODE is the master routine to specify a different performance item for analysis.

1300	1	38:3	722	BEGIN
1301	1	38:4	722	MSCREEN:=1;
1302	1	38:4	726	NPRINT:=1;
1303	1	38:4	730	CHANGEFUNCTIONALPURPOSES;
1304	1	38:3	732	END;
1305	1	38:2	732	NODE:=0;
1306	1	38:2	736	FOR I:=1 TO 300 DO
1307	1	38:3	752	IF CORE[I]=TEMP THEN
1308	1	38:4	783	BEGIN
1309	1	38:5	783	NODE:=I-1;
1310	1	38:4	791	END;
1311	1	38:2	801	IF NODE=0 THEN
1312	1	38:3	808	BEGIN
1313	1	38:4	808	WRITELN('Performance item # ',temp,' does not exist!');
1314	1	38:4	895	ANYKEY;
1315	1	38:3	897	END;
1316	1	38:1	897	UNTIL NODE<>0;
1317	1	38:0	904	END;
1318	1	38:0	922	

See previous page for program description.

1319	1	39:D	1	(#P#)PROCEDURE PRINTATTRIBUTES;
1320	1	39:0	0	BEGIN
1321	1	39:1	0	REWRITE(PRNT,'PRINTER:');
1322	1	39:1	21	TOPPAGE;
1323	1	39:1	23	PRINTTHEATTRIBUTES;
1324	1	39:1	25	CLOSE(PRNT);
1325	1	39:0	34	END;
1326	1	39:0	46	

PRINTATTRIBUTES controls printout of entire page of attributes.

1327 1 40:D 1 (\*P\*)PROCEDURE EXAMINEATTRIBUTES; FORWARD;  
1328 1 40:D 1

These procedures are presented later on in this program.

```

1329 1 41:D 1 (##P2)PROCEDURE PREPEXATTR;
1330 1 41:0 0 BEGIN
1331 1 41:1 0 REPEAT
1332 1 41:2 0 NODE:=NODE+1;
1333 1 41:2 8 IF CORE[NODE] DIV 10000 * 10000=CORE[NODE] THEN
1334 1 41:3 73 BEGIN
1335 1 41:4 73 NSCREEN:=1;
1336 1 41:4 77 NPRINT:=1;
1337 1 41:4 81 SEEK(DATANODE,CORE2[NODE]);
1338 1 41:4 105 GET(DATANODE);
1339 1 41:4 113 XOBJECTIVE:=DATANODE^.TAXA;
1340 1 41:4 123 NOBJECTIVE:=DATANODE^.NTAXAL2;
1341 1 41:3 138 END;
1342 1 41:2 138 IF (CORE[NODE] DIV 100 * 100 = CORE[NODE])
1343 1 41:2 197 AND(CORE[NODE] DIV 1000 * 1000<>CORE[NODE]) THEN
1344 1 41:3 263 BEGIN
1345 1 41:4 263 NSCREEN:=2;
1346 1 41:4 267 NPRINT:=2;
1347 1 41:4 271 SEEK(DATANODE,CORE2[NODE]);
1348 1 41:4 295 GET(DATANODE);
1349 1 41:4 303 XFUNPUR:=DATANODE^.TAXA;
1350 1 41:4 313 NFUNPUR:=DATANODE^.NTAXAL3;
1351 1 41:3 328 END;
1352 1 41:2 328 IF CORE[NODE] DIV 100 * 100 <> CORE[NODE] THEN
1353 1 41:3 389 BEGIN
1354 1 41:4 389 NSCREEN:=3;
1355 1 41:4 393 NPRINT:=3;
1356 1 41:4 397 SEEK(DATANODE,CORE2[NODE]);
1357 1 41:4 421 GET(DATANODE);
1358 1 41:4 429 XCHARAC:=DATANODE^.TAXA;
1359 1 41:4 439 NCHARAC:=DATANODE^.NTAXAL4;
1360 1 41:3 454 END;
1361 1 41:2 454 I:=TRUNC(CORE[NODE] DIV 1000000);
1362 1 41:1 508 UNTIL I<>0;
1363 1 41:1 515 PAC:=ASPECT[1];
1364 1 41:1 533 NPAC:=1;
1365 1 41:0 539 END;
1366 1 41:0 554

```

PREPEXATTR sets up header for an attributes analysis display.



```

1367 1 42:D 1 (%%P*)PROCEDURE PREEXAMINEATTRIBUTES;
1368 1 42:0 0 BEGIN
1369 1 42:1 0 NODE:=0;
1370 1 42:1 4 PREPKEY(109,'Would you like to begin analyzing the first performance item?');
1371 1 42:1 71 IF ANS='N' THEN
1372 1 42:2 78 CHANGENODE;
1373 1 42:1 80 PREPEXATTR;
1374 1 42:1 82 EXAMINEATTRIBUTES;
1375 1 42:0 84 END;
1376 1 42:0 96

```

PREEXAMINEATTRIBUTES prepares computer to process the first performance item.

```

1377 1 43:0 1 ($9P$)PROCEDURE EXMEAS;
1378 1 43:0 0 BEGIN
1379 1 43:1 0 IF OK THEN
1380 1 43:2 5 BEGIN
1381 1 43:3 5 REPEAT
1382 1 43:4 5 WRITE('Which one (type 0 to reconsider) ?');
1383 1 43:4 51 KEYN;
1384 1 43:4 53 IF (I<0) OR (I>NATTR) THEN
1385 1 43:5 68 BEGIN
1386 1 43:6 68 WRITELN('Please type an integer between 0 and ',NATTR,'. ');
1387 1 43:6 147 ANYKEY;
1388 1 43:5 149 END;
1389 1 43:3 149 UNTIL (I>=0) AND (I<=NATTR);
1390 1 43:3 164 IF I=0 THEN
1391 1 43:4 171 EXIT(EXMEAS);
1392 1 43:3 175 NCURATTRIBUTE:=ATTRINDEX[I];
1393 1 43:3 192 IF NCURATTRIBUTE>0 THEN
1394 1 43:4 199 EXAMINEMEASURES;
1395 1 43:2 201 END
1396 1 43:1 201 ELSE
1397 1 43:2 203 BEGIN
1398 1 43:3 203 GOTOXY(0,15);
1399 1 43:3 208 WRITE(CHR(11));
1400 1 43:3 218 WRITELN('There are no attributes to analyze');
1401 1 43:3 272 ANYKEY;
1402 1 43:2 274 END;
1403 1 43:0 274 END;
1404 1 43:0 290

```

EXMEAS asks analyst which measure he wishes to analyze.

```

1405 140:D 1 (##P*)PROCEDURE EXAMINEATTRIBUTES;
1406 140:0 0 BEGIN
1407 140:1 0 REPEAT
1408 140:2 0 TOPSCREEN;
1409 140:2 2 SHOWATTRIBUTES;
1410 140:2 4 GOTOXY(0,15);
1411 140:2 9 WRITELN(CHR(11));
1412 140:2 27 WRITE(' You may perform any of the following procedures:', chr(13),
1413 140:2 98 ' 1. Examine measures for an attribute      2. Add new attributes',chr(13),
1414 140:2 183 ' 3. Rerword an attribute          4. Remove an attribute',chr(13),
1415 140:2 269 ' 5. Print these attributes        6. Proceed to the NEXT perf item',chr(13),
1416 140:2 365 ' 7. Proceed to ANOTHER perf item  8. Select a different analytic proc',chr(13),
1417 140:2 464 'Please select one: ');
1418 140:2 495 REPEAT
1419 140:3 495 KEYN;
1420 140:3 497 IF (I<1) OR (I>8) THEN
1421 140:4 510 WRITELN('Please type an integer between 1 and 8');
1422 140:2 568 UNTIL (I>=1) AND (I<=8);
1423 140:2 581 CASE I OF
1424 140:2 586 1:EXMEAS;
1425 140:2 590 2:ADDATTRIBUTES;
1426 140:2 594 3:REWORDATTRIBUTES;
1427 140:2 598 4:DELETEATTRIBUTES;
1428 140:2 602 5:PRINTATTRIBUTES;
1429 140:2 606 6:PREPEXATTR;
1430 140:2 610 7:BEGIN
1431 140:4 610 CHANGENODE;
1432 140:4 612 PREPEXATTR;
1433 140:3 614 END;
1434 140:2 616 8:EXIT(EXAMINEATTRIBUTES);
1435 140:2 622 END;
1436 140:1 646 UNTIL NODE>=NCORELAST;
1437 140:1 655 WRITELN('All performance items have been processed');
1438 140:1 716 PREPKEY(99,'Do you wish to review any items?');
1439 140:1 754 IF ANS='N' THEN
1440 140:2 761 EXIT(EXAMINEATTRIBUTES);
1441 140:1 765 CHANGENODE;
1442 140:1 767 EXAMINEATTRIBUTES;
1443 140:0 769 END;
1444 140:0 786
1445 140:0 786 (##I 05:MEASATTR3.TEXT 8)
1446 140:0 786

```

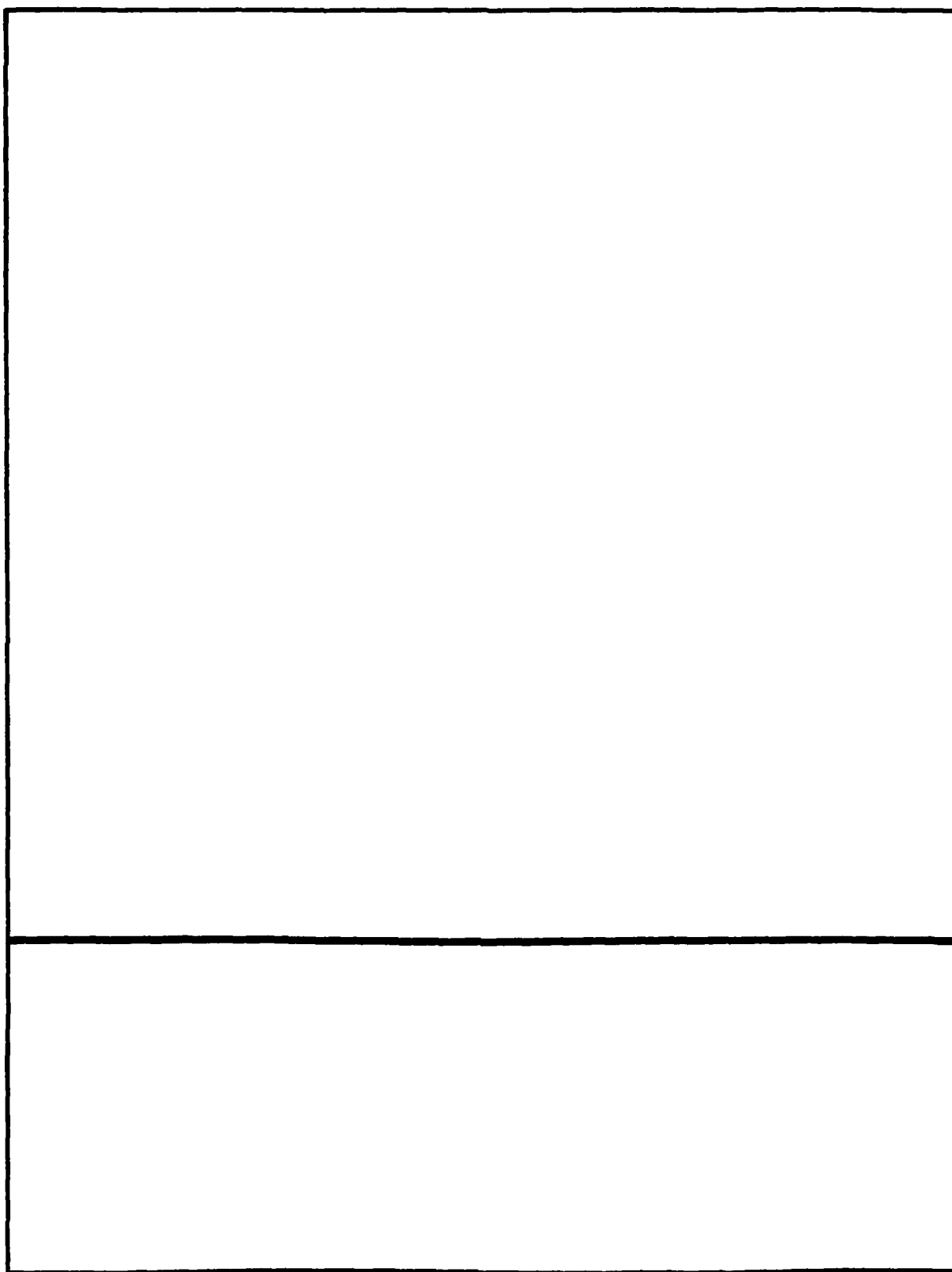
EXAMINEATTRIBUTES governs setting up an entire display for examining attributes.

```

1447 1110 0($P$)BEGIN
1448 1110 0 ($N-2)
1449 1111 0 INLINECALL:=0;
1450 1111 93 NMEASURES:=400;
1451 1111 99 NATTRIBUTES:=200;
1452 1111 105 BRANCHIN;
1453 1111 107 DEFINEASPECTS;
1454 1111 110 APMSK:=CONCAT(COPY(CURSYS,1,2),(COPY(CURSP,1,2)),COPY(CURSUB,1,2),'');
1455 1111 202 NAMEATCORE:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)), 'AC');
1456 1111 305 NAMEATTRIBUTES:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,
1,4)), 'AT');
1457 1111 408 NAMECORE:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)), 'MC');
1458 1111 511 NAMEMEASURES:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,
4)), 'ME');
1459 1111 614 CORENAME:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)), 'CO');
1460 1111 717 DATANAME:=CONCAT(APMSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)), 'FI');
1461 1111 820 OPENDATAFILE;
1462 1111 823 OPENATTRIBUTESFILE;
1463 1111 826 OPENMEASURESFILE;
1464 1111 829 READATTRFILE;
1465 1111 832 READMEASFILE;
1466 1111 835 READCOREFILE;
1467 1111 838 SORTCOREFILE;
1468 1111 841 PREEXAMINEATTRIBUTES;
1469 1111 843 CLOSEATTRFILE;
1470 1111 845 CLOSEMEASFILE;
1471 1111 847 BRANCHOUT;
1472 1111 849 SETCHAIN('GREETING');
1473 1110 863 END.

```

MAINROUTINE for specifying attributes and measures.



### **MEASUREMENT PURPOSE PROGRAM (MEASPURP)**

The measurement purpose program allows the analyst to edit measurement purposes (adding, rewording and deleting as appropriate). It also allows the analyst to associate (or disassociate) measurement purposes with characteristics. As characteristics are associated and disassociated, the corresponding objectives and functional purposes are treated in a similar way. Thus, objectives and functional purposes are never associated (or disassociated) directly.

```

1 1 1:D 1 ($L PRINTER: 8)
2 1 1:D 1 ($S+)
3 1 1:D 1 ($ Program to compose measurement purpose index)
4 1 1:D 1 ($ Ronald G. Shapiro Version 2.0 10/25/82)
5 1 1:D 1 Program Formissue;
6 1 1:D 3
7 28 1:D 3
8 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
9 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
10 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
11 28 5:D 1 PROCEDURE SWAPON;
12 28 6:D 1 PROCEDURE SWAPOFF;
13 28 6:D 1
14 1 1:D 1 Uses Chainstuff;
15 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

16 1 1:D 3 ($P*)TYPE
17 1 1:D 3 PASSFILE =RECORD
18 1 1:D 3 CURSYS,CURSP,CURSUB,PAC:STRING(80);
19 1 1:D 3 NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
20 1 1:D 3 END;
21 1 1:D 3
22 1 1:D 3 DATABASE=RECORD
23 1 1:D 3 NTAXA: ARRAY[1..4] OF INTEGER;
24 1 1:D 3 TAXA: STRING(80);
25 1 1:D 3 END;
26 1 1:D 3
27 1 1:D 3 ISSUEFILE =RECORD
28 1 1:D 3 NUM:INTEGER;
29 1 1:D 3 NAME: ARRAY[1..2] OF STRING(80);
30 1 1:D 3 DATA: ARRAY[1..225] OF INTEGER(8);
31 1 1:D 3 END;
32 1 1:D 3
33 1 1:D 3 FASTFILE=RECORD
34 1 1:D 3 PRINTIT:PACKED ARRAY[1..300]OF BOOLEAN;
35 1 1:D 3 END;
36 1 1:D 3

```

PASSFILE for communication between programs [see GREETING program].  
 DATABASE contains a basic list of performance items. ISSUEFILE contains  
 measurement purpose names and references to performance items. FASTFILE  
 allows for fast printout of repeated performance items.



```

37 1 1:D 3 (*P*)VAR
38 1 1:D 3 DATANODE:FILE OF DATABASE;
39 1 1:D 348 COREFILE:FILE OF INTEGER(8);
40 1 1:D 651 PASSNODE:FILE OF PASSFILE;
41 1 1:D 1122 ISSUE:FILE OF ISSUEFILE;
42 1 1:D 2180 FASTISSUE: FILE OF FASTFILE;
43 1 1:D 2499
44 1 1:D 2499 XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING(80);
45 1 1:D 2786 NCURISSUE,NCCHARAC,NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
46 1 1:D 2794
47 1 1:D 2794 APHDSK:STRING(8);
48 1 1:D 2799 NAMEFASTISSUE,CORENAME,DATANAME,ISSUENAME: STRING(24);
49 1 1:D 2851
50 1 1:D 2851 CORE: ARRAY[1..300] OF INTEGER(8);
51 1 1:D 3751 CORE2: ARRAY[1..300] OF INTEGER;
52 1 1:D 4051
53 1 1:D 4051 ISSUEDATA: ARRAY[1..225] OF INTEGER(8);
54 1 1:D 4726
55 1 1:D 4726 FLAG: ARRAY[1..300] OF BOOLEAN;
56 1 1:D 5026
57 1 1:D 5026 ASPECT: ARRAY[1..5] OF STRING(14);
58 1 1:D 5066 INVERSEA: ARRAY[1..5] OF INTEGER;
59 1 1:D 5071
60 1 1:D 5071 DISPCOUNT,GOPAGE,COUNT,INVERSE,HELP,NSCREEN,MODE:INTEGER;
61 1 1:D 5078 NCORELAST,NISSUES,NUISSUES,ITEMCOUNT:INTEGER;
62 1 1:D 5082 TEMPL1,TEMPL2,TEMPL3,TEMPL4,TEMPL5,TEMPL6,TEMP,CORELAST: INTEGER(8);
63 1 1:D 5106 I,J,K,L,M,N,NN,INLINECALL,INDENT,NLENGTH,LLENGTH,PC,TEMP2:INTEGER;
64 1 1:D 5119
65 1 1:D 5119 CHARACTERISTIC,NEXTCHARACTERISTIC,LASTCHARACTERISTIC:BOOLEAN;
66 1 1:D 5122 REFERENCED,LONGWAY,DONE,OVER,OK,SKIP:BOOLEAN;
67 1 1:D 5128
68 1 1:D 5128 LINER:STRING(80);
69 1 1:D 5169 LINE:STRING(80);
70 1 1:D 5210 PROCESS:STRING(15);
71 1 1:D 5218
72 1 1:D 5218 ANSWER,REGLINE:STRING(80);
73 1 1:D 5300
74 1 1:D 5300 ANS,ANSOLD: CHAR;
75 1 1:D 5302
76 1 2:D 1 PROCEDURE EXAMINEISSUES;FORWARD;
77 1 3:D 1 PROCEDURE COMPACTISSUES;FORWARD;
78 1 3:D 1
78 1 3:D 1 (*I 05:UTILITY.TEXT*)
79 1 3:D 1

```

These strings, arrays and variables are used by this program.

```

80 1 4:D 1 ($Ps)PROCEDURE ANYKEY;
81 1 4:0 0 BEGIN
82 1 4:1 0 WRITELN(' ');
83 1 4:1 18 WRITELN('*** Please press any key to continue ***');
84 1 4:1 78 ($R-8)
85 1 4:1 78 READ(ANS);
86 1 4:1 89 ($R+8)
87 1 4:0 89 END;
88 1 4:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

89 1 5:D 1 (%%P%)PROCEDURE HELPER;
90 1 5:0 0 BEGIN
91 1 5:1 0 WRITELN('For help please refer to your APM MANUAL.');
```

```

92 1 5:0 61 END;
93 1 5:0 74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```

94 1 6:D 1 (86P)PROCEDURE KEYN;
95 1 6:D 1 VAR
96 1 6:D 1 ANSWER: STRING[40];
97 1 6:D 22 II: ARRAY[1..4] OF INTEGER;
98 1 6:D 26 OK: BOOLEAN;
99 1 6:D 27 IIO: INTEGER;
100 1 6:D 28
101 1 6:D 0 BEGIN
102 1 6:D 0 (86R-8)
103 1 6:1 0 REPEAT
104 1 6:2 0 REPEAT
105 1 6:3 0 ANSWER:= '
106 1 6:3 27 OK:=TRUE;
107 1 6:3 30 READLN(ANSWER);
108 1 6:3 49 IF LENGTH(ANSWER)=0 THEN
109 1 6:4 57 WRITELN('Please enter the integer again');
110 1 6:2 107 UNTIL LENGTH(ANSWER)>0;
111 1 6:2 115 IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
112 1 6:3 130 HELPER;
113 1 6:2 132 FOR I:=1 TO 4 DO
114 1 6:3 147 BEGIN
115 1 6:4 147 I[I]:=ORD(ANSWER[I])-48;
116 1 6:4 165 IF (I[I]<0) OR (I[I]>9) THEN
117 1 6:5 192 BEGIN
118 1 6:6 192 IF (I=1) OR (I[I]<>(ORD(' ')-48)) THEN
119 1 6:7 214 BEGIN
120 1 6:8 214 OK:=FALSE;
121 1 6:8 217 WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
122 1 6:7 275 END;
123 1 6:5 275 END;
124 1 6:3 275 END;
125 1 6:1 285 UNTIL OK=TRUE;
126 1 6:1 292 IIO:=I[I];
127 1 6:1 302 FOR I:=2 TO 4 DO
128 1 6:2 317 BEGIN
129 1 6:3 317 IF (I[I]>=0) AND (I[I]<=9) THEN
130 1 6:4 344 IIO:=IIO*10+I[I];
131 1 6:2 361 END;
132 1 6:2 371 (86R+8)
133 1 6:1 371 I:=IIO;
134 1 6:D 376 END;
135 1 6:D 398

```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive integers, KEYN will display an appropriate warning and wait for a response.

```

136 1 7:D 1 (86P8)PROCEDURE KEY;
137 1 7:D 1 VAR
138 1 7:D 1 II2:INTEGER;
139 1 7:0 0 BEGIN
140 1 7:0 0 (86R-8)
141 1 7:1 0 ANSWER:= ' ;
142 1 7:1 27 REPEAT
143 1 7:2 27 READLN(ANSWER);
144 1 7:2 47 ANS:=ANSWER[1];
145 1 7:2 55 IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
146 1 7:2 78 (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27) THEN
147 1 7:3 98 WRITELN('PLEASE RESPOND YES OR NO!');
148 1 7:2 143 IF (ORD(ANS)>90) THEN
149 1 7:3 150 BEGIN
150 1 7:4 150 II2:=ORD(ANS)-32;
151 1 7:4 157 ANS:=CHR(II2);
152 1 7:3 161 END;
153 1 7:1 161 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
154 1 7:1 186 (86R+8)
155 1 7:1 186 IF ANS='H' THEN
156 1 7:2 193 HELPER;
157 1 7:0 195 END;
158 1 7:0 210

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

159 1 0:D 1 (80P8)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
160 1 0:0 0 BEGIN
161 1 0:1 0 HELP:=HLP;
162 1 0:1 9 REPEAT
163 1 0:2 9 WRITE(MSG);
164 1 0:2 20 KEY;
165 1 0:1 22 UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
166 1 0:0 41 END;
167 1 0:0 56

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

168 1 9:D 1 (SSP)PROCEDURE INLINE;
169 1 9:D 1 VAR
170 1 9:D 1 LONGLINE:STRING[125];
171 1 9:D 64 LINEOK:BOOLEAN;
172 1 9:D 63
173 1 9:0 0 BEGIN
174 1 9:1 0 REPEAT
175 1 9:2 0 READLN(LONGLINE);
176 1 9:2 19 LINEOK:=TRUE;
177 1 9:2 22 M:=LENGTH(LONGLINE);
178 1 9:2 29 IF M>68 THEN
179 1 9:3 36 BEGIN
180 1 9:4 36 WRITELN('WARNING LINE CONTAINS OVER 68 CHARACTERS');
181 1 9:4 100 WRITELN(' ');
182 1 9:4 118 PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 68 CHARACTERS? ');
183 1 9:4 166 IF ANS='N' THEN
184 1 9:5 173 BEGIN
185 1 9:6 173 LINEOK:=FALSE;
186 1 9:6 176 WRITELN('PLEASE TYPE LINE AGAIN: ');
187 1 9:5 220 END
188 1 9:4 220 ELSE
189 1 9:5 222 M:=68;
190 1 9:3 226 END;
191 1 9:1 226 UNTIL LINEOK;
192 1 9:1 230 INLINECALL:=INLINECALL+1;
193 1 9:1 238 IF INLINECALL>25 THEN
194 1 9:2 245 BEGIN
195 1 9:3 245 WRITELN('WARNING--You have typed in over 25 new attributes and/or',
196 1 9:3 313 chr(13),' measures--the limit for the demonstration. Please select',
197 1 9:3 394 chr(13),' a different analytic procedure before entering more data',
198 1 9:3 474 chr(13),' --or risk losing everything you have done today!');
199 1 9:3 554 ANYKEY;
200 1 9:2 556 END;
201 1 9:1 556 LINER:=COPY(LONGLINE,1,M);
202 1 9:0 574 END;
203 1 9:0 592

```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```

204 1 10:0 1 (800)PROCEDURE SHOWALINE;
205 1 10:0 0 BEGIN
206 1 10:1 0 NLENGTH:=LENGTH(LINE);
207 1 10:1 0 WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
208 1 10:2 24 NLENGTH:=NLENGTH-1;
209 1 10:1 36 IF NLENGTH<2 THEN
210 1 10:2 43 EXIT(SHOWALINE);
211 1 10:1 47 IF NLENGTH<=LLENGTH THEN
212 1 10:2 56 BEGIN
213 1 10:3 56 WRITE(LINE);
214 1 10:3 68 EXIT(SHOWALINE);
215 1 10:2 72 END;
216 1 10:1 72 L:=LLENGTH;
217 1 10:1 78 WHILE (LINE[L]<>' ') AND (L>1) DO
218 1 10:2 96 L:=L-1;
219 1 10:1 104 L:=L-1;
220 1 10:1 114 IF L>0 THEN
221 1 10:2 121 BEGIN
222 1 10:3 121 REGLINE:=COPY(LINE,1,L);
223 1 10:3 140 WRITELN(REGLINE);
224 1 10:2 160 END;
225 1 10:1 160 L:=L+2;
226 1 10:1 168 NLENGTH:=NLENGTH-L+1;
227 1 10:1 180 IF NLENGTH<1 THEN
228 1 10:2 187 EXIT(SHOWALINE);
229 1 10:1 191 REGLINE:=COPY(LINE,L,NLENGTH);
230 1 10:1 212 FOR I:=1 TO INDENT DO
231 1 10:2 228 WRITE(' ');
232 1 10:1 248 WRITE(REGLINE);
233 1 10:1 260 PC:=PC+1;
234 1 10:0 268 END;
235 1 10:0 284

```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.



```

236 1 11:0 1 (80P8)PROCEDURE BRANCHIN;
237 1 11:0 0 BEGIN
238 1 11:0 0 (80I-8)
239 1 11:1 0 RESET(PASSNODE,'PASSTHRU');
240 1 11:1 19 I:=IORESULT;
241 1 11:1 24 (80I+8)
242 1 11:1 24 IF I<>0 THEN
243 1 11:2 31 BEGIN
244 1 11:3 31 WRITELN('PASSTHRU FILE DOES NOT EXIST');
245 1 11:3 79 WRITELN(' #####FATAL ERROR#####');
246 1 11:3 124 WRITELN(' ',I);
247 1 11:3 168 ANYKEY;
248 1 11:3 170 SETCHAIN('PGM1');
249 1 11:3 180 EXIT(PROGRAM);
250 1 11:2 184 END;
251 1 11:1 184 GET(PASSNODE);
252 1 11:1 192 CURSYS:=PASSNODE^.CURSYS;
253 1 11:1 202 CURSP:=PASSNODE^.CURSP;
254 1 11:1 212 CURSUB:=PASSNODE^.CURSUB;
255 1 11:1 222 PAC:=PASSNODE^.PAC;
256 1 11:1 230 NCURSYS:=PASSNODE^.NCURSYS;
257 1 11:1 239 NCURSP:=PASSNODE^.NCURSP;
258 1 11:1 248 NCURSUB:=PASSNODE^.NCURSUB;
259 1 11:1 257 NPAC:=PASSNODE^.NPAC;
260 1 11:1 264 CLOSE(PASSNODE);
261 1 11:0 275 END;
262 1 11:0 290

```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```

263 1 12:0 1 (80P)PROCEDURE BRANCHOUT;
264 1 12:0 0 BEGIN
265 1 12:1 0 REWRITE(PASSNODE,'PASSTHRU');
266 1 12:1 21 PASSNODE^.FLAG1:=1;
267 1 12:1 29 PUT(PASSNODE);
268 1 12:1 37 CLOSE(PASSNODE,LOCK);
269 1 12:0 44 END;
270 1 12:0 58
271 1 12:0 58
272 1 12:0 58
273 1 12:0 58 (80I 05:UTILITY.TEXT8)
274 1 12:0 58

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```

275 1 13:D 1 (80P)PROCEDURE DEFINEASPECTS;
276 1 13:0 0 BEGIN
277 1 13:1 0 ASPECT[1]='Potentialities';
278 1 13:1 30 ASPECT[2]='Processes';
279 1 13:1 55 ASPECT[3]='Products';
280 1 13:1 79 ASPECT[4]='Environment';
281 1 13:1 106 ASPECT[5]='Constraints';
282 1 13:0 133 END;
283 1 13:0 146

```

DEFINEASPECTS assigns names to each aspect.

```

284 1 1418 1 ($P$)PROCEDURE READCOREFILE;
285 1 1410 0 BEGIN
286 1 1410 0 ($I-8)
287 1 1411 0 RESET(COREFILE,CORENAME);
288 1 1411 11 I:=IORESULT;
289 1 1411 16 ($I+8)
290 1 1411 16 IF I<>0 THEN
291 1 1412 23 BEGIN
292 1 1413 23 WRITELN('COREFILE DOES NOT EXIST');
293 1 1413 66 WRITELN(' ****FATAL ERROR**** ');
294 1 1413 109 WRITELN(' ',I);
295 1 1413 152 ANYKEY;
296 1 1413 154 BRANCHOUT;
297 1 1413 156 SETCHAIN('GREETING');
298 1 1413 170 EXIT(PROGRAM);
299 1 1412 174 END
300 1 1411 174 ELSE
301 1 1412 176 FOR I:=1 TO 300 DO
302 1 1413 192 BEGIN
303 1 1414 192 GET(COREFILE);
304 1 1414 200 CORE[I]:=COREFILE^;
305 1 1413 228 END;
306 1 1411 238 GET(COREFILE);
307 1 1411 246 CORELAST:=COREFILE^;
308 1 1411 262 NCORELAST:=TRUNC(CORELAST);
309 1 1411 275 CLOSE(COREFILE)
310 1 1410 284 END;
311 1 1410 300

```

READCOREFILE reads performance item index file from disk into core.

```

312 1 15:0 1 (86P8)PROCEDURE SORTCOREFILE;
313 1 15:0 0 BEGIN
314 1 15:1 0 FOR I:=1 TO 300 DO
315 1 15:2 16 CORE2[I]:=I;
316 1 15:1 45 I:=2;
317 1 15:1 49 REPEAT
318 1 15:2 49 IF CORE[I]<CORE[I-1] THEN
319 1 15:3 94 BEGIN
320 1 15:4 94 TEMP:=CORE[I];
321 1 15:4 122 CORE[I]:=CORE[I-1];
322 1 15:4 164 CORE[I-1]:=TEMP;
323 1 15:4 194 TEMP2:=CORE2[I];
324 1 15:4 213 CORE2[I]:=CORE2[I-1];
325 1 15:4 247 CORE2[I-1]:=TEMP2;
326 1 15:4 268 IF I>2 THEN
327 1 15:5 275 I:=I-1;
328 1 15:3 283 END
329 1 15:2 283 ELSE
330 1 15:3 285 I:=I+1;
331 1 15:1 293 UNTIL I>NCORELAST;
332 1 15:0 302 END;
333 1 15:0 320

```

SORTCOREFILE constructs the permutation vector for the performance items.

```

334 1 16:D 1 ($P$)PROCEDURE OPENISSUEINDEX;
335 1 16:0 0 BEGIN
336 1 16:0 0 ($I-$)
337 1 16:1 0 RESET(ISSUE,ISSUENAME);
338 1 16:1 11 ($I+$)
339 1 16:1 11 IF IORESULT<>0 THEN
340 1 16:2 17 BEGIN
341 1 16:3 17 WRITELN('Please bear with me while I create the Issue Index on the disk');
342 1 16:3 99 REWRITE(ISSUE,ISSUENAME);
343 1 16:3 112 FOR I:=1 TO 225 DO
344 1 16:4 128 ISSUE^.DATA[I]:=0;
345 1 16:3 167 FOR I:=1 TO NISSUES DO
346 1 16:4 183 BEGIN
347 1 16:5 183 FOR J:=1 TO 2 DO
348 1 16:6 197 ISSUE^.NAME[J]:='
349 1 16:5 268 ISSUE^.NUM:=I;
350 1 16:5 275 SEEK(ISSUE,I);
351 1 16:5 286 PUT(ISSUE);
352 1 16:5 294 IF (EOF(ISSUE))THEN
353 1 16:6 304 BEGIN
354 1 16:7 304 WRITELN('OUT OF DISK SPACE');
355 1 16:7 341 WRITELN(' **FATAL ERROR** ');
356 1 16:7 378 ANYKEY;
357 1 16:7 380 SETCHAIN('GREETING');
358 1 16:7 394 EXIT(PROGRAM);
359 1 16:6 398 END;
360 1 16:4 398 END;
361 1 16:3 408 CLOSE(ISSUE,LOCK);
362 1 16:3 417 OPENISSUEINDEX;
363 1 16:3 419 EXIT(OPENISSUEINDEX);
364 1 16:2 423 END
365 1 16:1 423 ELSE
366 1 16:2 425 BEGIN
367 1 16:3 425 NUISSUES:=NISSUES+1;
368 1 16:3 433 REPEAT
369 1 16:4 433 NUISSUES:=NUISSUES-1;
370 1 16:4 441 SEEK(ISSUE,NUISSUES);
371 1 16:4 452 GET(ISSUE)
372 1 16:3 460 UNTIL (ISSUE^.NAME[1]<>' ' ) OR (NUISSUES=1);
373 1 16:3 527 IF (NUISSUES=1) AND (COPY(ISSUE^.NAME[1],1,5)=' ' ) THEN
374 1 16:4 567 NUISSUES:=0;
375 1 16:2 571 END;
376 1 16:0 571 END;
377 1 16:0 598

```

OPENISSUEINDEX counts how many measurement purposes were specified in previous analyses. If ISSUE file does not exist, it creates one.

```

378 1 17:0 1 ($P*)PROCEDURE DISPLAYNAME;
379 1 17:0 0 BEGIN
380 1 17:1 0 SEEK(ISSUE,I);
381 1 17:1 11 GET(ISSUE);
382 1 17:1 19 WRITELN(I,'. ',ISSUE^.NAME[1],CHR(13),' ',ISSUE^.NAME[2],CHR(13));
383 1 17:0 135 END;
384 1 17:0 148

```

DISPLAYNAME displays the name of one measurement purpose.

```

385 1 18:D 1 (%P*)PROCEDURE DISPLAYISSUES;
386 1 18:0 0 BEGIN
387 1 18:1 0 PAGE(OUTPUT);
388 1 18:1 10 IF NUISSUES=0 THEN
389 1 18:2 17 BEGIN
390 1 18:3 17 WRITELN('Currently, there are no measurement purposes in the APM for this
system and subsystem');
391 1 18:2 146 END
392 1 18:1 146 ELSE
393 1 18:2 148 BEGIN
394 1 18:3 148 WRITELN('The following measurement purposes are currently included ');
395 1 18:3 238 FOR I:=1 TO NUISSUES DO
396 1 18:4 254 BEGIN
397 1 18:5 254 DISPLAYNAME;
398 1 18:5 256 IF (I MOD 6=0) THEN
399 1 18:6 265 BEGIN
400 1 18:7 265 ANYKEY;
401 1 18:7 267 PAGE(OUTPUT);
402 1 18:6 277 END;
403 1 18:4 277 END;
404 1 18:2 287 END;
405 1 18:0 287 END;
406 1 18:0 306

```

DISPLAYISSUES displays names of all measurement purposes.



```

407 1 19:D 1 (38P)PROCEDURE ERASEFASTISSUE(III:INTEGER);
408 1 19:0 0 BEGIN;
409 1 19:1 0 RESET(FASTISSUE,NAMEFASTISSUE);
410 1 19:1 13 IF IORESULT= 0 THEN
411 1 19:2 19 BEGIN
412 1 19:3 19 SEEK(FASTISSUE,III);
413 1 19:3 28 FOR J:=1 TO 300 DO
414 1 19:4 44 FASTISSUE^.PRINTIT[J]:=FALSE;
415 1 19:3 72 PUT(FASTISSUE);
416 1 19:3 80 CLOSE(FASTISSUE);
417 1 19:2 89 END;
418 1 19:0 89 END;
419 1 19:0 104

```

ERASEFASTISSUE: FASTISSUE must be erased for any measurement purpose being modified. ERASEFASTISSUE does this erasure.

```

420 1 20:D 1 (%%P%)PROCEDURE ADDISSUE(AI:INTEGER);
421 1 20:0 0 BEGIN
422 1 20:1 0 SEEK(ISSUE,AI);
423 1 20:1 9 GET(ISSUE);
424 1 20:1 17 WRITELN('Please describe the new measurement purpose in 2 68-character lines');
425 1 20:1 104 FOR I:=1 TO 2 DO
426 1 20:2 118 BEGIN
427 1 20:3 118 WRITELN('Please type line #',I,'');
428 1 20:3 178 REPEAT
429 1 20:4 178 INLINE;
430 1 20:4 180 IF LENGTH(LINER)>68 THEN
431 1 20:5 189 WRITELN('Line contains over 68 characters, please retype');
432 1 20:3 256 UNTIL LENGTH(LINER)<=68;
433 1 20:3 265 ISSUE^.NAME[I]:=LINER;
434 1 20:2 285 END;
435 1 20:1 295 SEEK(ISSUE,AI);
436 1 20:1 304 PUT(ISSUE);
437 1 20:0 312 END;
438 1 20:0 330

```

ADDISSUE adds a measurement purpose.

```

439 1 21:0 1 (80P)PROCEDURE REMOVEISSUE(RI:INTEGER);
440 1 21:0 0 BEGIN
441 1 21:1 0 SEEK(ISSUE,RI);
442 1 21:1 9 GET(ISSUE);
443 1 21:1 17 FOR J:=1 TO 2 DO
444 1 21:2 31 ISSUE^.NAME[J]:='
445 1 21:1 102 FOR J:=1 TO 225 DO
446 1 21:2 118 ISSUE^.DATA[J]:=0;
447 1 21:1 157 SEEK (ISSUE,RI);
448 1 21:1 166 PUT(ISSUE);
449 1 21:1 174 IF RI=NUISSUES THEN
450 1 21:2 181 NUISSUES:=NUISSUES-1;
451 1 21:1 189 ERASEFASTISSUE(RI);
452 1 21:0 192 END;
453 1 21:0 208

```

REMOVEISSUE removes a measurement purpose from the measurement purpose list.

```

454 1 22:0 1 (80P)PROCEDURE ALTERISSUES;
455 1 22:0 0 BEGIN
456 1 22:1 0 DISPLAYISSUES;
457 1 22:1 2 GOTOXY(0,16);
458 1 22:1 7 WRITE(CHR(11));
459 1 22:1 17 WRITE('You may perform any of the following procedures:',chr(13),
460 1 22:1 87 ' 1. Analyze a measurement purpose 2. Specify a new measurement purpose',chr(13),
461 1 22:1 181 ' 3. Remove a measurement purpose 4. Replace a measurement purpose',chr(13),
462 1 22:1 271 ' 5. Pack meas purposes efficiently 6. Select a different analytic proc.',chr(13),
463 1 22:1 365 'Please select one: ');
464 1 22:1 396 REPEAT
465 1 22:2 396 KEYN;
466 1 22:2 398 IF (I<1) OR (I>6) THEN
467 1 22:3 411 Writeln('Please type an integer between 1 and 5');
468 1 22:1 469 UNTIL (I>=1) AND (I<=6);
469 1 22:1 482 CASE I OF
470 1 22:1 487 1: EXAMINEISSUES;
471 1 22:1 491 2: BEGIN
472 1 22:3 491 GOTOXY(0,16);
473 1 22:3 496 WRITE(CHR(11));
474 1 22:3 506 IF NUISSUES>=NUISSUES THEN
475 1 22:4 515 BEGIN
476 1 22:5 515 Writeln('ISSUE INDEX IS FULL--NO ADDITIONAL ISSUES CAN BE ADDED');
477 1 22:5 589 ANYKEY;
478 1 22:4 591 END
479 1 22:3 591 ELSE
480 1 22:4 593 BEGIN
481 1 22:5 593 NUISSUES:=NUISSUES+1;
482 1 22:5 601 ADISSUE(NUISSUES);
483 1 22:4 606 END;
484 1 22:2 606 END;
485 1 22:1 608 3: BEGIN
486 1 22:3 608 WRITE('Which one (type 0 when done)? ');
487 1 22:3 650 KEYN;
488 1 22:3 652 IF I<>0 THEN
489 1 22:4 659 REMOVEISSUE(I);
490 1 22:2 664 END;
491 1 22:1 666 4: BEGIN
492 1 22:3 666 WRITE('Which one (type 0 when done)?');
493 1 22:3 707 KEYN;

```

ALTERISSUES presents menu of options showing what analyst can do with measurement purposes.

494	1	22:3	709	IF I<>0 THEN
495	1	22:4	716	BEGIN
496	1	22:5	716	GOTOXY(0,16);
497	1	22:5	721	WRITE(CMR(11));
498	1	22:5	731	PREPKEY(73,'Is this merely an improvement in the descriptor? ');
499	1	22:5	798	IF ANS='Y' THEN
500	1	22:6	805	BEGIN
501	1	22:7	805	ADDISSUE(I);
502	1	22:6	810	END
503	1	22:5	810	ELSE
504	1	22:6	812	BEGIN
505	1	22:7	812	REMOVEISSUE(I);
506	1	22:7	817	NUISSUES:=NUISSUES+1;
507	1	22:7	825	ADDISSUE(I);
508	1	22:6	830	END;
509	1	22:4	830	END;
510	1	22:2	830	END;
511	1	22:1	832	5: COMPACTISSUES;
512	1	22:1	836	6: BEGIN
513	1	22:3	836	BRANCHOUT;
514	1	22:3	838	SETCHAIN('GREETING');
515	1	22:3	852	EXIT(PROGRAM);
516	1	22:2	856	END;
517	1	22:1	858	END;
518	1	22:1	878	ALTERISSUES;
519	1	22:0	880	END;
520	1	22:0	898	

See previous page for program description.

```

521 1 23:0 1 (80%)PROCEDURE GETINDEX;
522 1 23:0 0 BEGIN
523 1 23:1 0 PAGE(OUTPUT);
524 1 23:1 10 WRITELN('Please be patient...',chr(13),' I am preparing the computer for you');
525 1 23:1 109 BEGIN
526 1 23:2 109 SEEK(ISSUE,NCURISSUE);
527 1 23:2 120 GET(ISSUE);
528 1 23:2 128 FOR I:=1 TO 225 DO
529 1 23:3 144 BEGIN
530 1 23:4 144 ISSUEDATA[I]:=ISSUE^.DATA[I];
531 1 23:3 186 END;
532 1 23:1 196 END;
533 1 23:0 196 END;
534 1 23:0 210

```

GETINDEX places reference to performance item into array ISSUEDATA or the measurement purpose currently being processed.

```

535 1 24:D 1 (80P)PROCEDURE OPENDATAFILE;
536 1 24:0 0 BEGIN
537 1 24:0 0 (80I-8)
538 1 24:1 0 RESET(DATANODE,DATANAME);
539 1 24:1 10 (80I+8)
540 1 24:1 10 I:=IORESULT;
541 1 24:1 15 IF I<>0 THEN
542 1 24:2 22 BEGIN
543 1 24:3 22 WRITELN('DATABASE MUST BE CREATED BEFORE ISSUES ARE LINKED TO DATABASE');
544 1 24:3 103 ANYKEY;
545 1 24:3 105 BRANCHOUT;
546 1 24:3 107 SETCHAIN('GREETING');
547 1 24:3 121 EXIT(PROGRAM);
548 1 24:2 125 END;
549 1 24:0 125 END;
550 1 24:0 138

```

OPENDATAFILE verifies the presence of performance items.

```

551 1 25:0 1 (%P*)PROCEDURE TOPSCREEN;
552 1 25:0 0 BEGIN
553 1 25:1 0 GOTOXY(0,4);
554 1 25:1 5 WRITE(CHR(11));
555 1 25:1 15 NSCREEN:=3;
556 1 25:1 19 M:=LENGTH(CURSYS);
557 1 25:1 27 IF M>16 THEN
558 1 25:2 34 M:=16;
559 1 25:1 38 LINE:=COPY(CURSYS,1,M);
560 1 25:1 57 WRITE('s',LINE,' Systems');
561 1 25:1 99 GOTOXY(26,4);
562 1 25:1 104 M:=LENGTH(CURSP);
563 1 25:1 112 IF M>16 THEN
564 1 25:2 119 M:=16;
565 1 25:1 123 LINE:=COPY(CURSP,1,M);
566 1 25:1 142 WRITE('s',LINE);
567 1 25:1 164 GOTOXY(44,4);
568 1 25:1 169 M:=LENGTH(CURSUB);
569 1 25:1 177 IF M>16 THEN
570 1 25:2 184 M:=16;
571 1 25:1 188 LINE:=COPY(CURSUB,1,M);
572 1 25:1 207 WRITELN('s',LINE);
573 1 25:1 237 GOTOXY(62,4);
574 1 25:1 242 WRITELN('s',PAC);
575 1 25:1 272 M:=LENGTH(XOBJECTIVE);
576 1 25:1 280 IF M>67 THEN M:=67;
577 1 25:1 291 LINE:=COPY(XOBJECTIVE,1,M);
578 1 25:1 310 IF NSCREEN>1 THEN
579 1 25:2 317 WRITELN('Objective[',NOBJECTIVE,']:',LINE);
580 1 25:1 385 M:=LENGTH(XFUNPUR);
581 1 25:1 393 IF M>67 THEN M:=67;
582 1 25:1 404 LINE:=COPY(XFUNPUR,1,M);
583 1 25:1 423 IF NSCREEN>2 THEN
584 1 25:2 430 WRITELN('Fct1 Prps[',NFUNPUR,']:',LINE);
585 1 25:1 498 WRITELN(' ');
586 1 25:0 516 END;
587 1 25:0 528

```

TOPSCREEN produces the header material on the display screen.



```

588 1 26:0 1  (##P#)PROCEDURE SAVEINDEX;
589 1 26:0 0  BEGIN
590 1 26:1 0  PAGE(OUTPUT);
591 1 26:1 10 WRITELN('Please be patient...',chr(13),' I am saving all of your hard work');
592 1 26:1 108 SEEK(ISSUE,NCURISSUE);
593 1 26:1 119 GET(ISSUE);
594 1 26:1 127 FOR I:=1 TO 225 DO
595 1 26:2 143 BEGIN
596 1 26:3 143 ISSUE^.DATA[I]:=ISSUEDATA[I];
597 1 26:2 185 END;
598 1 26:1 195 SEEK(ISSUE,NCURISSUE);
599 1 26:1 206 PUT(ISSUE);
600 1 26:0 214 END;
601 1 26:0 228

```

SAVEINDEX records the references to performance items for a given measurement purpose in the issuedata file for use by other programs.

```

602 1 3:0 1 (*P*)PROCEDURE COMPACTISSUES;
603 1 3:0 0 BEGIN
604 1 3:1 0 FOR J:=1 TO NISSUES DO
605 1 3:2 16 ERASEFASTISSUE(J);
606 1 3:1 31 M:=0;
607 1 3:1 35 I:=0;
608 1 3:1 39 REPEAT
609 1 3:2 39 I:=I+1;
610 1 3:2 47 REPEAT
611 1 3:3 47 J:=I+M;
612 1 3:3 57 SEEK(ISSUE,J);
613 1 3:3 68 GET(ISSUE);
614 1 3:3 76 IF (COPY(ISSUE^.NAME[1],1,5)=' ' THEN
615 1 3:4 110 M:=M+1;
616 1 3:2 118 UNTIL (COPY(ISSUE^.NAME[1],1,5)<>' ' OR (J>NISSUES));
617 1 3:2 160 IF J<=NISSUES THEN
618 1 3:3 169 BEGIN
619 1 3:4 169 J:=I+M;
620 1 3:4 179 SEEK(ISSUE,I);
621 1 3:4 190 PUT(ISSUE);
622 1 3:3 198 END;
623 1 3:1 198 UNTIL J>=NISSUES;
624 1 3:1 207 I:=I+1;
625 1 3:1 215 FOR K:=1 TO NISSUES DO
626 1 3:2 233 REMOVEISSUE(K);
627 1 3:1 248 IF M>0 THEN NISSUES:=NISSUES-M+1;
628 1 3:0 267 END;
629 1 3:0 288 (*I 05:MEASPURP2.TEXT*)
630 1 3:0 288

```

COMPACTISSUES packs measurement purpose references more efficiently.

```

631 1 27:0 1 ($P*)PROCEDURE REFISSUE;
632 1 27:0 0 BEGIN
633 1 27:1 0 J:=0;
634 1 27:1 4 REPEAT
635 1 27:2 4 J:=J+1;
636 1 27:1 12 UNTIL(ISSUEDATA[J]=0) OR (J=224);
637 1 27:1 50 IF (J=224) AND (ISSUEDATA[J]<>0) THEN
638 1 27:2 88 BEGIN
639 1 27:3 88 WRITELN('SORRY--BUT YOU ALREADY HAVE 224 REFERENCES FOR THIS MEASUREMENT
PURPOSE',CH R(13),
' SO YOU CAN NOT ADD ANOTHER ONE!!!');
640 1 27:3 181 ANYKEY;
641 1 27:3 236 EXIT(REFISSUE)
642 1 27:3 238 END
643 1 27:2 242 ELSE
644 1 27:1 242 BEGIN
645 1 27:2 244 ISSUEDATA[J]:=CORE[I];
646 1 27:3 244 FLAG[I]:=TRUE;
647 1 27:3 284 END;
648 1 27:2 301 END;
649 1 27:0 301 END;
650 1 27:0 318

```

REFISSUE adds a new performance item reference to the measurement purpose index.

```

651 1 28:0 1 (#P8)PROCEDURE UNREFISSUE;
652 1 28:0 0 BEGIN
653 1 28:1 0 K:=0;
654 1 28:1 4 REPEAT
655 1 28:2 4 K:=K+1;
656 1 28:1 12 UNTIL (K=224) OR (ISSUEDATA[K]=CORE[I]);
657 1 28:1 63 IF K>=224 THEN
658 1 28:2 72 BEGIN
659 1 28:3 72 WRITELN('ERROR--FLAG SAYS REFERENCED, ISSUEDATA SAYS UNREFERENCED');
660 1 28:3 148 EXIT(UNREFISSUE);
661 1 28:2 152 END;
662 1 28:1 152 J:=K-1;
663 1 28:1 160 REPEAT
664 1 28:2 160 J:=J+1;
665 1 28:2 168 ISSUEDATA[J]:=ISSUEDATA[J+1];
666 1 28:1 210 UNTIL (ISSUEDATA[J]=0) OR (J=224);
667 1 28:1 248 ISSUEDATA[225]:=0;
668 1 28:1 275 FLAG[I]:=FALSE;
669 1 28:0 292 END;
670 1 28:0 308

```

UNREFISSUE removes a reference to a performance item from a measurement purpose index.

```

671 1 29:D 1 (SPR)PROCEDURE REWORD;
672 1 29:0 0 BEGIN
673 1 29:1 0 ADDISSUE(NCURISSUE);
674 1 29:0 5 END;
675 1 29:0 18

```

REWORD allows one to reword a measurement purpose label.

```

676 1 30:0 1 (##P)PROCEDURE SETUPFLAG;
677 1 30:0 0 BEGIN
678 1 30:1 0 FOR K:=1 TO 300 DO
679 1 30:2 16 FLAG[K]:=FALSE;
680 1 30:1 43 IF ISSUEDATA[I]<>0 THEN
681 1 30:2 71 BEGIN
682 1 30:3 71 WRITELN('Please be patient...', chr(13),
683 1 30:3 113 ' I am setting up your measurement purpose');
684 1 30:3 175 FOR K:=1 TO 225 DO
685 1 30:4 191 BEGIN
686 1 30:5 191 IF ISSUEDATA[K]<>0 THEN
687 1 30:6 221 BEGIN
688 1 30:7 221 NODE:=0;
689 1 30:7 225 REPEAT
690 1 30:8 225 NODE:=NODE+1;
691 1 30:8 233 IF ISSUEDATA[K]=CORE[NODE] THEN
692 1 30:9 274 FLAG[NODE]:=TRUE;
693 1 30:7 293 UNTIL (NODE=300);
694 1 30:6 302 END;
695 1 30:4 302 END;
696 1 30:2 312 END;
697 1 30:0 312 END;
698 1 30:0 332

```

SETUPFLAG sets up a flag for each performance item which belongs to a measurement purpose.

```

699 1 31:D 1 (89P8)PROCEDURE GOEXAMINE;
700 1 31:0 0 BEGIN
701 1 31:1 0 OK:=FALSE;
702 1 31:1 4 REPEAT
703 1 31:2 4 GOTOXY(0,16);
704 1 31:2 9 WRITE(CHR(11),'Which one would you like to analyze(type 0 to reconsider)?');
705 1 31:2 89 KEYN;
706 1 31:2 91 NCURISSUE:=I;
707 1 31:2 97 IF (I>NUISSUES) OR (I<0) THEN
708 1 31:3 112 WRITELN('Please type an integer between 1 and ',NUISSUES,'!');
709 1 31:1 191 UNTIL (I<=NUISSUES) AND (I>=0);
710 1 31:1 206 IF I=0 THEN
711 1 31:2 213 EXIT(EXAMINEISSUES);
712 1 31:1 217 ERASEFASTISSUE(I);
713 1 31:1 222 GETINDEX;
714 1 31:1 224 SETUPFLAG;
715 1 31:1 226 PAGE(OUTPUT);
716 1 31:1 236 WRITELN('You have chosen to analyze measurement purpose: ',NCURISSUE);
717 1 31:1 316 I:=NCURISSUE;
718 1 31:1 322 DISPLAYNAME;
719 1 31:1 324 GOTOXY(0,3);
720 1 31:1 329 WRITELN(CHR(26),'3Black on white',chr(26),
721 1 31:1 376 '2 performance items are associated with the measurement purpose');
722 1 31:1 459 RESET(DATANODE,DATANAME);
723 1 31:0 471 END;
724 1 31:0 486

```

GOEXAMINE determines which measurement purpose the analyst wishes to analyze.

```

725 1 32:D 1 ($00)PROCEDURE REVERSEISSUES;
726 1 32:0 0 BEGIN
727 1 32:1 0 IF FLAG(I)=TRUE
728 1 32:1 16 THEN UNREFISSUE
729 1 32:1 21 ELSE REFISSUE;
730 1 32:1 27 COUNT:=0;
731 1 32:0 31 END;
732 1 32:0 44

```

REVERSEISSUES—if analyst wishes to add a performance item to the measurement purpose, reverse issues calls REFISSUE. If analyst wishes to remove a performance item from the measurement purpose, REVERSEISSUES calls UNREFISSUE.



```

733 1 33:0 1 (##P*)PROCEDURE ENDPAGE;
734 1 33:0 0 BEGIN
735 1 33:1 0 I:=0;
736 1 33:1 4 GOTOXY(0,19);
737 1 33:1 9 WRITE(CHR(11));
738 1 33:1 19 PREPKEY(94,'Change assns between measurement purpose and a performance item?');
739 1 33:1 95 IF ORD(ANS)=27 THEN
740 1 33:2 102 BEGIN
741 1 33:3 102 CLOSE(DATANODE);
742 1 33:3 110 SAVEINDEX;
743 1 33:3 112 EXIT(EXAMINEISSUES);
744 1 33:2 116 END;
745 1 33:1 116 IF ANS='Y' THEN
746 1 33:2 123 BEGIN
747 1 33:3 123 GOTOXY(0,19);
748 1 33:3 128 WRITE(CHR(11));
749 1 33:3 138 WRITE('Which one (type 0 if none; 999 if all)? ');
750 1 33:3 190 KEYN;
751 1 33:3 192 I:=I+GOPAGE-1;
752 1 33:3 204 IF (I>0) AND (I<300) THEN
753 1 33:4 219 IF (CHARACTERISTIC=TRUE) THEN
754 1 33:5 227 REVERSEISSUES
755 1 33:4 227 ELSE
756 1 33:5 231 BEGIN
757 1 33:6 231 GOTOXY(0,21);
758 1 33:6 236 WRITE(CHR(11));
759 1 33:6 246 WRITELN('ERROR--PERFORMANCE ITEM ',I,' IS NOT A CHARACTERISTIC!');
760 1 33:5 338 END;
761 1 33:3 338 N:=999+GOPAGE-1;
762 1 33:3 350 IF I=N THEN
763 1 33:4 359 FOR I:=GOPAGE TO NODE DO
764 1 33:5 377 IF (CORE[I]-CORE[I] DIV 100 * 100<>0) THEN
765 1 33:6 447 REVERSEISSUES;
766 1 33:3 459 NODE:=GOPAGE-1;
767 1 33:2 467 END
768 1 33:1 467 ELSE
769 1 33:2 469 GOPAGE:=NODE+1;
770 1 33:1 477 GOTOXY(0,8);
771 1 33:1 482 WRITE(CHR(11));
772 1 33:0 492 END;
773 1 33:0 508

```

ENDPAGE displays the "do you want to change association" message and then it processes the response.

```

774 1 34:D 1 ($$P$)PROCEDURE CHANGETOPSCREEN;
775 1 34:0 0 BEGIN
776 1 34:1 0 IF CORE[NODE]=0 THEN
777 1 34:2 30 EXIT(CHANGETOPSCREEN);
778 1 34:1 34 SEEK(DATANODE,CORE2[NODE]);
779 1 34:1 57 GET(DATANODE);
780 1 34:1 64 IF CORE[NODE] DIV 10000 * 10000 = CORE[NODE] THEN
781 1 34:2 129 BEGIN
782 1 34:3 129 I:=TRUNC(CORE[NODE] DIV 1000000);
783 1 34:3 183 PAC:=ASPECT[I];
784 1 34:3 201 NPAC:=I;
785 1 34:3 207 XOBJECTIVE:=DATANODE^.TAXA;
786 1 34:3 215 NOBJECTIVE:=DATANODE^.NTAXA[2];
787 1 34:2 228 END
788 1 34:1 228 ELSE
789 1 34:2 230 BEGIN
790 1 34:3 230 XFUPPUR:=DATANODE^.TAXA;
791 1 34:3 238 NFUPPUR:=DATANODE^.NTAXA[3];
792 1 34:2 251 END;
793 1 34:1 251 IF(NEXTCHARACTERISTIC=TRUE) THEN
794 1 34:2 259 BEGIN
795 1 34:3 259 TOPSCREEN;
796 1 34:3 261 COUNT:=0;
797 1 34:3 265 GOTOXY(0,8);
798 1 34:3 270 WRITE(CHR(11));
799 1 34:2 280 END;
800 1 34:0 280 END;
801 1 34:0 292

```

CHANGETOPSCREEN changes contents of the header printed at the top of each page.

```

002 1 35:0 1 ($$P$)PROCEDURE ONEPERFITEMDISPLAY;
003 1 35:0 0 BEGIN
004 1 35:1 0 SEEK(DATANODE,CORE2(NODE));
005 1 35:1 23 GET(DATANODE);
006 1 35:1 30 K:=DATANODE^.NTAXA[N];
007 1 35:1 45 LLENGTH:=72;
008 1 35:1 49 LINE:=DATANODE^.TAXA;
009 1 35:1 57 IF K<>0 THEN
010 1 35:2 64 BEGIN
011 1 35:3 64 INVERSE:=2;
012 1 35:3 68 NN:=0;
013 1 35:3 72 CASE N OF
014 1 35:3 77 1:TEMPL5:=1000000;
015 1 35:3 114 2:TEMPL5:=10000;
016 1 35:3 133 3:TEMPL5:=100;
017 1 35:3 150 4:TEMPL5:=1;
018 1 35:3 167 END;
019 1 35:3 182 TEMPL6:=CORE(NODE) DIV TEMPL5;
020 1 35:3 220 REPEAT
021 1 35:4 220 NN:=NN+1;
022 1 35:3 228 UNTIL (NN=225) OR (ISSUEDATA[NN] DIV TEMPL5 = TEMPL6);
023 1 35:3 277 IF NN<225 THEN
024 1 35:4 286 INVERSE:=3;
025 1 35:3 290 WRITE(' ',CHR(26),INVERSE,K,'. ');
026 1 35:3 352 INDENT:=6;
027 1 35:3 356 SHOWALINE;
028 1 35:3 358 WRITELN(CHR(26),'2');
029 1 35:2 386 END;
030 1 35:0 386 END;
031 1 35:0 402

```

ONEPERFITEMDISPLAY displays one performance item in the body of the display used to select which performance item ought to be processed next.

```

032 1 36:0 1 (88P8)PROCEDURE SHOWPERFITEMS;
033 1 36:0 0 BEGIN
034 1 36:1 0 OK:=FALSE;
035 1 36:1 4 DISPCOUNT:=0;
036 1 36:1 8 IF M=2 THEN TEMPL1:=1000000;
037 1 36:1 50 IF M=2 THEN TEMPL3:=10000;
038 1 36:1 74 IF M=3 THEN TEMPL1:=10000;
039 1 36:1 98 IF M=3 THEN TEMPL3:=100;
040 1 36:1 120 TEMPL2:=TEMP DIV TEMPL1;
041 1 36:1 146 FOR NODE:=1 TO NCORELAST DO
042 1 36:2 162 BEGIN
043 1 36:3 162 IF (TEMPL2=CORE[NODE] DIV TEMPL1) AND
044 1 36:3 201 (CORE[NODE] DIV TEMPL3 * TEMPL3 = CORE[NODE]) THEN
045 1 36:4 265 BEGIN
046 1 36:5 265 OK:=TRUE;
047 1 36:5 269 ONEPERFITEMDISPLAY;
048 1 36:5 271 DISPCOUNT:=DISPCOUNT+1;
049 1 36:4 279 END;
050 1 36:3 279 IF (DISPCOUNT DIV 15 * 15=DISPCOUNT) AND (DISPCOUNT<>0) THEN
051 1 36:4 298 BEGIN
052 1 36:5 298 DISPCOUNT:=0;
053 1 36:5 302 ANYKEY;
054 1 36:5 304 GOTOXY(0,2);
055 1 36:5 309 WRITE(CHR(11));
056 1 36:4 319 END;
057 1 36:2 319 END;
058 1 36:1 329 IF OK=FALSE THEN
059 1 36:2 337 WRITELN(' ...none');
060 1 36:0 367 END;
061 1 36:0 384

```

SHOWPERFITEMS controls production of the body of displays of performance items.

```

862 1 37:D 1 (89P)PROCEDURE SPECIFYSTART;
863 1 37:0 0 BEGIN
864 1 37:1 0 REPEAT
865 1 37:2 0 GOTOXY(0,4);
866 1 37:2 5 WRITE(CHR(11));
867 1 37:2 15 Writeln('The following aspects are part of the APM:');
868 1 37:2 77 FOR N:=1 TO 5 DO
869 1 37:3 91 INVERSEA[N]:=2;
870 1 37:2 116 FOR NODE:=1 TO 300 DO
871 1 37:3 132 IF FLAG[NODE]=TRUE THEN
872 1 37:4 153 BEGIN
873 1 37:5 153 N:=TRUNC(CORE[NODE] DIV 1000000);
874 1 37:5 207 IF N>0 THEN
875 1 37:6 214 INVERSEA[N]:=3;
876 1 37:4 229 END;
877 1 37:2 239 FOR N:=1 TO 5 DO
878 1 37:3 253 Writeln(' ',CHR(26),INVERSEA[N],N,' ',ASPECT[N],CHR(26),'2');
879 1 37:2 382 WRITE('Please select one: ');
880 1 37:2 413 KEYN;
881 1 37:2 415 PAC:=ASPECT[I];
882 1 37:2 433 NPAC:=I;
883 1 37:2 439 TEMPL4:=I;
884 1 37:2 456 TEMP:=TEMPL4*1000000;
885 1 37:2 501 GOTOXY(0,4);
886 1 37:2 506 WRITE(CHR(11));
887 1 37:2 516 N:=2;
888 1 37:2 520 Writeln('The following Objectives are available for the aspect selected: ');
889 1 37:2 604 SHOWPERFITEMS;
890 1 37:2 606 WRITE('Please select one: ');
891 1 37:2 637 KEYN;
892 1 37:2 639 NOBJECTIVE:=I;
893 1 37:2 645 TEMPL4:=I;
894 1 37:2 662 TEMP:=TEMP+TEMPL4*10000;
895 1 37:2 699 FOR I:=1 TO 300 DO
896 1 37:3 715 IF TEMP=CORE[I] THEN
897 1 37:4 746 BEGIN
898 1 37:5 746 SEEK(DATANODE,CORE2[NODE]);
899 1 37:5 769 GET(DATANODE);
900 1 37:5 776 NOBJECTIVE:=DATANODE^.TAXA;
901 1 37:4 784 END;

```

SPECIFYSTART allows analyst to select where he/she wants to start analyzing measurement purpose links to performance items.

```

902 1 37:2 794      GOTOXY(0,4);
903 1 37:2 799      WRITE(CHR(11));
904 1 37:2 809      N:=3;
905 1 37:2 813      WRITELN('The following Functional Purposes are available for the objective
                    selected: ');
906 1 37:2 909      SHOWPERFITEMS;
907 1 37:2 911      WRITE('Please select one: ');
908 1 37:2 942      KEYN;
909 1 37:2 944      TEMPL4:=1;
910 1 37:2 961      TEMP:=TEMP+TEMPL4*100;
911 1 37:2 996      FOR I:=1 TO 300 DO
912 1 37:3 1012        IF CORE[I]=TEMP THEN
913 1 37:4 1043          BEGIN
914 1 37:5 1043            NODE:=I-1;
915 1 37:5 1051            GOPAGE:=NODE+1;
916 1 37:4 1059          END;
917 1 37:2 1069      IF NODE=0 THEN
918 1 37:3 1076        WRITELN('Performance item # ',temp,' does not exist!');
919 1 37:1 1163      UNTIL NODE<>0;
920 1 37:0 1170      END;
921 1 37:0 1196

```

See previous page for program description.

```

922 1 2:D 1 (SOP*)PROCEDURE EXAMINEISSUES;
923 1 2:0 0 BEGIN
924 1 2:1 0 GOEXAMINE;
925 1 2:1 2 IF OK=TRUE THEN
926 1 2:2 10 EXIT(EXAMINEISSUES);
927 1 2:1 14 LASTCHARACTERISTIC:=TRUE;
928 1 2:1 18 IF CORE[NCORELAST]-CORE[NCORELAST] DIV 100 * 100 = 0 THEN
929 1 2:2 88 LASTCHARACTERISTIC:=FALSE;
930 1 2:1 92 NODE:=0;
931 1 2:1 96 GOPAGE:=1;
932 1 2:1 100 PREPKEY(222,'Do you wish to analyze the first performance item?');
933 1 2:1 158 IF ANS='N' THEN
934 1 2:2 165 SPECIFYSTART;
935 1 2:1 167 REPEAT
936 1 2:2 167 REPEAT
937 1 2:3 167 NODE:=NODE+1;
938 1 2:3 175 CHARACTERISTIC:=TRUE;
939 1 2:3 179 IF CORE[NODE]-CORE[NODE] DIV 100 * 100 = 0 THEN
940 1 2:4 249 CHARACTERISTIC:=FALSE;
941 1 2:3 253 NEXTCHARACTERISTIC:=TRUE;
942 1 2:3 257 IF CORE[NODE+1]-CORE[NODE+1] DIV 100 * 100 = 0 THEN
943 1 2:4 331 NEXTCHARACTERISTIC:=FALSE;
944 1 2:3 335 IF (CHARACTERISTIC=FALSE) THEN
945 1 2:4 343 CHANGETOPSCREEN;
946 1 2:3 345 IF CHARACTERISTIC=FALSE THEN
947 1 2:4 353 GOPAGE:=GOPAGE+1;
948 1 2:2 361 UNTIL (CHARACTERISTIC=TRUE) OR (NODE>=NCORELAST);
949 1 2:2 377 IF (NCORELAST<=GOPAGE) AND (LASTCHARACTERISTIC=FALSE)
950 1 2:2 390 THEN BEGIN
951 1 2:4 393 CLOSE (DATANODE);
952 1 2:4 401 SAVEINDEX;
953 1 2:4 403 EXIT(EXAMINEISSUES);
954 1 2:3 407 END;
955 1 2:2 407 INVERSE:=2;
956 1 2:2 411 SEEK(DATANODE,CORE2[NODE]);
957 1 2:2 434 GET(DATANODE);
958 1 2:2 441 IF FLAG[NODE]=TRUE THEN
959 1 2:3 462 INVERSE:=3;
960 1 2:2 466 L:=NODE-GOPAGE+1;
961 1 2:2 478 WRITE(CHR(26),INVERSE,' ',L,'. ');

```

EXAMINEISSUES does initial setup for analyzing measurement purpose-performance item links.

```

942 1 2:2 542 FOR K:=1 TO 4 DO
943 1 2:3 556 BEGIN
944 1 2:4 556 J:=DATANODE^.NTAXA[K];
945 1 2:4 571 WRITE(J,',');
946 1 2:3 593 END;
947 1 2:2 603 WRITE(' ');
948 1 2:2 613 LINE:=DATANODE^.TAXA;
949 1 2:2 621 WRITE(' ');
970 1 2:2 631 LLENGTH:=60;
971 1 2:2 635 INDENT:=14;
972 1 2:2 639 SHOWALINE;
973 1 2:2 641 WRITELN(CHR(26),'2');
974 1 2:2 669 COUNT:=COUNT+1;
975 1 2:2 677 IF (COUNT=5) OR (NODE=NCORELAST) OR
976 1 2:2 690 (CORE[NODE] DIV 100<>CORE[NODE+1] DIV 100) THEN
977 1 2:3 754 ENDPAGE;
978 1 2:1 756 UNTIL (I=0) AND (NODE=NCORELAST);
979 1 2:1 771 CLOSE(DATANODE);
980 1 2:1 779 SAVEINDEX;
981 1 2:0 781 END;
982 1 2:0 800
983 1 2:0 800
984 1 2:0 800
985 1 2:0 800
986 1 2:0 800 (##) 05:NEASPURP2.TEXT#)
987 1 2:0 800

```

See previous page for program description.



```

988 1:0      ($$P$)BEGIN
989 1:0      0 ($$N+8)
990 1:1      0 INLINECALL:=0;
991 1:1      62 NISSUES:=5;
992 1:1      66 BRANCHIN;
993 1:1      68 DEFINEASPECTS;
994 1:1      70 APHDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
995 1:1      162 CORENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
996 1:1      265 NAMEFASTISSUE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FA');
997 1:1      368 WRITELN('I am now sorting your performance items.');
```

```

998 1:1      428 READCOREFILE;
999 1:1      430 SORTCOREFILE;
1000 1:1      432 ISSUENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
1001 1:1      535 DATANAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
1002 1:1      638 OPENISSUEINDEX;
1003 1:1      640 ALTERISSUES;
1004 1:0      642 END.

```

Main Program: reads and sorts core file, the index to the performance items. Then, control is transformed to the ALTERISSUES program which presents the list of measurement pruposes and the various analytic procedures which may be performed with measurement purposes.

## **PRINT**

PRINT allows the analyst to print either 1) all performance items, attributes and measures, or 2) performance items, attributes and measures for a given measurement purpose. The analyst may choose to print only some objectives, functional purposes and characteristics in this program without altering the basic data set.

```

1 1 1:D 1 ($L PRINTER: 8)
2 1 1:D 1 ($S+8)
3 1 1:D 1 ($ Program to print performance items, attribute, and measures list for a given
    measurement purpo
4 1 1:D 1 ($ Ronald G. Shapiro Version 2.0 10/25/82$)
5 1 1:D 1 Program Printdatasetts;
6 1 1:D 3
7 28 1:D 3
8 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
9 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
10 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
11 28 5:D 1 PROCEDURE SWAPON;
12 28 6:D 1 PROCEDURE SWAPOFF;
13 28 6:D 1
14 22 1:D 1
15 22 1:D 3
16 22 2:D 3 FUNCTION PADDLE(SELECT: INTEGER): INTEGER;
17 22 3:D 3 FUNCTION BUTTON(SELECT: INTEGER): BOOLEAN;
18 22 4:D 1 PROCEDURE TTLOUT(SELECT: INTEGER; DATA: BOOLEAN);
19 22 5:D 3 FUNCTION KEYPRESS: BOOLEAN;
20 22 6:D 3 FUNCTION RANDOM: INTEGER;
21 22 7:D 1 PROCEDURE RANDOMIZE;
22 22 8:D 1 PROCEDURE NOTE(PITCH,DURATION: INTEGER);
23 22 8:D 3
24 1 1:D 3 Uses Chainstuff,APPLESTUFF;
25 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

26 1 1:D 3 ($P$)CONST
27 1 1:D 3 OBJBL1='The system must be capable of:;'
28 1 1:D 3 OBJBL2='The system must carry out the following activities:;'
29 1 1:D 3 OBJBL3='The system must produce:;'
30 1 1:D 3 OBJBL4='Performance objectives must be met despite:;'
31 1 1:D 3 OBJBL5='Performance objectives must be met despite:;'
32 1 1:D 3

```

CONSTANTS are defined.

```

33 1 1:D 3 (###)TYPE
34 1 1:D 3 ISSUEFILE =RECORD
35 1 1:D 3   MUN:INTEGER;
36 1 1:D 3   NAME:ARRAY[1..2]OF STRING[80];
37 1 1:D 3   DATA:ARRAY[1..225]OF INTEGER[8];
38 1 1:D 3   END;
39 1 1:D 3
40 1 1:D 3 FASTFILE =RECORD
41 1 1:D 3   PRINTIT:PACKED ARRAY[1..300]OF BOOLEAN;
42 1 1:D 3   END;
43 1 1:D 3
44 1 1:D 3 PASSFILE =RECORD
45 1 1:D 3   CURSYS,CURSP,CURSUB,PAC:STRING[80];
46 1 1:D 3   NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
47 1 1:D 3   END;
48 1 1:D 3
49 1 1:D 3 DATABASE =RECORD
50 1 1:D 3   NTAXA: ARRAY[1..4] OF INTEGER;
51 1 1:D 3   TAXA: STRING[80];
52 1 1:D 3   END;
53 1 1:D 3
54 1 1:D 3 FILEATTRIBUTES =RECORD
55 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
56 1 1:D 3   DESCRIPTOR: STRING[68];
57 1 1:D 3   END;
58 1 1:D 3
59 1 1:D 3 FILEMEASURES =RECORD
60 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
61 1 1:D 3   DESCRIPTOR: STRING[68];
62 1 1:D 3   END;
63 1 1:D 3

```

ISSUEFILE is a list of measurement purpose names and references to performance items. FASTFILE allows fast processing of measurement purposes. PASSFILE is an inter-program communication. DATABASE is performance item files. FILEATTRIBUTES is attributes file. FILEMEASURES is measures file.

```

64 1 11D 3 (80PS)VAR
65 1 11D 3 PASSNODE:FILE OF PASSFILE;
66 1 11D 474 DATANODE:FILE OF DATABASE;
67 1 11D 819 COREFILE:FILE OF INTEGER(8);
68 1 11D 1122 ATTRIBUTES:FILE OF FILEATTRIBUTES;
69 1 11D 1463 ATTRFILE:FILE OF INTEGER(12);
70 1 11D 1767 MEASURES:FILE OF FILEMEASURES;
71 1 11D 2108 MEASFILE:FILE OF INTEGER(12);
72 1 11D 2412 ISSUE:FILE OF ISSUEFILE;
73 1 11D 3470 FASTISSUE:FILE OF FASTFILE;
74 1 11D 3789
75 1 11D 3789 CORE:ARRAY[1..300] OF INTEGER(8);
76 1 11D 4689 ATTRCORE:ARRAY[1..200] OF INTEGER(12);
77 1 11D 5489 MEASCORE:ARRAY[1..400] OF INTEGER(12);
78 1 11D 7089 ASPECT:ARRAY[1..5] OF STRING(20);
79 1 11D 7144 CORE2:ARRAY[1..300] OF INTEGER;
80 1 11D 7444 ATTR2:ARRAY[1..200] OF INTEGER;
81 1 11D 7644 MEAS2:ARRAY[1..400] OF INTEGER;
82 1 11D 8044 PRINTIT:PACKED ARRAY[1..300] OF BOOLEAN;
83 1 11D 8063
84 1 11D 8063 XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING(80);
85 1 11D 8309 NCURMEASURE,NCURATTRIBUTE,NCURISSUE,
86 1 11D 8309 MFUNPUR,NBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
87 1 11D 8318
88 1 11D 8318 ISSUENAME,NAMEATCORE,NAMEATTRIBUTES,NAMEHCORE,NAMEMEASURES: STRING(30);
89 1 11D 8398 NAMEFASTISSUE,CORENAME,DATANAME: STRING(30);
90 1 11D 8446 LEVEL: STRING(10);
91 1 11D 8452 APHDSK:STRING(10);
92 1 11D 8458 USERNAME,USERDATE,USERMSG: STRING(80);
93 1 11D 8581
94 1 11D 8581 TEMP,CORELAST,T1,T2,T3,T4,T5: INTEGER(8);
95 1 11D 8602 TEMPX,ATTRLAST,MEASLAST:INTEGER(12);
96 1 11D 8614
97 1 11D 8614 MODE,INVERSE,HELP,NSCREEN:INTEGER;
98 1 11D 8618 NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
99 1 11D 8621 NISSUES,NUISSUES,NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;
100 1 11D 8626
101 1 11D 8626 I,J,K,L,M,N,CUT,INDENT,COUNT,TEMP2:INTEGER;
102 1 11D 8636
103 1 11D 8636 NOISSUE,REFERENCED,LONGWAY,DONE,OVER,OK,SKIP,NONE:BOOLEAN;

```

These strings, arrays and variables are used by this program.

```

104 1 1:D 8644
105 1 1:D 8644 LINER:STRING[4];(8ADDED TO AVOID COMPILER ERROR ON INLINE--NOT USED IN PRINT PGS)
106 1 1:D 8647 ANSWER,LINE:STRING[80];
107 1 1:D 8729
108 1 1:D 8729 ANS,ANSOLD: CHAR;
109 1 1:D 8731
110 1 1:D 8731 PRNT:TEXT;
111 1 1:D 9032
112 1 2:D 1 PROCEDURE ANYKEY;FORWARD;
113 1 3:D 1 PROCEDURE BRANCHIN;FORWARD;
114 1 4:D 1 PROCEDURE BRANCHOUT;FORWARD;
115 1 5:D 1 PROCEDURE ELIMINATE;FORWARD;
116 1 5:D 1

```

Continuation of strings, arrays and variables list from previous page.

```

117 7 110 1 (000)SEGMENT PROCEDURE OPENISSUEINDEX;
118 7 110 0 BEGIN
119 7 111 0 NOISSUE:=FALSE;
120 7 111 4 (001-8)
121 7 111 4 RESET(ISSUE,ISSUENAME);
122 7 111 15 (001+8)
123 7 111 15 IF IORESULT<>0 THEN
124 7 112 21 BEGIN
125 7 113 21 WRITELN('NO MEAS PURP FILE!');
126 7 113 59 NUISSUES:=0;
127 7 113 63 NOISSUE:=TRUE;
128 7 112 67 END
129 7 111 67 ELSE
130 7 112 69 BEGIN
131 7 113 69 NUISSUES:=NUISSUES+1;
132 7 113 77 REPEAT
133 7 114 77 NUISSUES:=NUISSUES-1;
134 7 114 85 SEEK(ISSUE,NUISSUES);
135 7 114 96 GET(ISSUE);
136 7 113 104 UNTIL(COPY(ISSUE^.NAME(1,1,5)<>' ' ) OR (NUISSUES=1);
137 7 113 144 IF (NUISSUES=1) AND (COPY(ISSUE^.NAME(1,1,5)=' ' ) THEN
138 7 114 184 NUISSUES:=0;
139 7 112 188 END;
140 7 111 188 CLOSE(ISSUE);
141 7 110 197 END;
142 7 110 212

```

OPENISSUEINDEX determines how many measurement purposes there are (if any).



```

143 0 1:D 1 ($P$)SEGMENT PROCEDURE TERMINATE;
144 0 1:0 0 BEGIN
145 0 1:1 0 WRITELN('PLEASE RUN PROC02 TO CREATE ATTRIB & MEAS');
146 0 1:1 61 ANYKEY;
147 0 1:1 64 BRANCHOUT;
148 0 1:1 67 SETCHAIN('GREETING');
149 0 1:1 81 EXIT(PROGRAM);
150 0 1:0 95 END;
151 0 1:0 98

```

TERMINATE displays warning message and transfers control back to analytic procedure section of GREETING program.

```

152 9 11D 1 (80P8)SEGMENT PROCEDURE OPENFASTISSUE;
153 9 110 0 BEGIN
154 9 110 0 (801-8)
155 9 111 0 RESET(FASTISSUE,NAMEFASTISSUE);
156 9 111 11 (801+8)
157 9 111 11 I:=IORESULT;
158 9 111 14 IF I<>0 THEN
159 9 112 23 BEGIN
160 9 113 23 REWRITE(FASTISSUE,NAMEFASTISSUE);
161 9 113 34 FOR I:=1 TO NISSUES DO
162 9 114 32 BEGIN
163 9 115 32 SEEK(FASTISSUE,I);
164 9 115 63 FOR J:=1 TO 300 DO
165 9 116 79 FASTISSUE^.PRINTIT(J):=FALSE;
166 9 115 107 PUT(FASTISSUE);
167 9 115 115 IF (EOF(FASTISSUE))THEN
168 9 116 125 BEGIN
169 9 117 125 WRITELN('OUT OF DISK SPACE');
170 9 117 162 WRITELN(' **FATAL ERROR** ');
171 9 117 199 ANYKEY;
172 9 117 202 BRANCHOUT;
173 9 117 205 SETCHAIN('GREETING');
174 9 117 219 EXIT(PROGRAM);
175 9 116 223 END;
176 9 114 223 END;
177 9 113 233 CLOSE(FASTISSUE,LOCK);
178 9 113 242 OPENFASTISSUE;
179 9 113 244 EXIT(OPENFASTISSUE);
180 9 112 248 END;
181 9 111 248 CLOSE(FASTISSUE);
182 9 110 257 END;
183 9 110 278

```

OPENFASTISSUE determines whether fastissue file exists. If not, it creates it.

```

184 10 118 1 (80P)SEGMENT PROCEDURE READATTRFILE;
185 10 110 0 BEGIN
186 10 110 0 (801-8)
187 10 111 0 RESET(ATTRFILE,NAMEATCORE);
188 10 111 11 I:=IORESULT;
189 10 110 16 (80I+8);
190 10 111 16 IF I<>0 THEN
191 10 112 23 TERMINATE
192 10 111 23 ELSE
193 10 112 28 BEGIN
194 10 113 28 FOR I:=1 TO NATTRIBUTES DO
195 10 114 44 BEGIN
196 10 115 44 GET(ATTRFILE);
197 10 115 52 ATTRCORE(I):=ATTRFILE^;
198 10 114 80 END;
199 10 113 90 GET(ATTRFILE);
200 10 113 98 ATTRLAST:=ATTRFILE^;
201 10 113 114 NATTRLAST:=TRUNC(ATTRLAST);
202 10 113 127 CLOSE(ATTRFILE);
203 10 112 136 END;
204 10 110 136 END;
205 10 110 150

```

READATTRFILE loads core with index to attributes file.

```

204 11 118 1 (80P)SEGMENT PROCEDURE READMEASFILE;
207 11 110 0 BEGIN
208 11 110 0 (80I-8)
209 11 111 0 RESET(MEASFILE,NAMENECORE);
210 11 111 11 I:=IORESULT;
211 11 110 16 (80I+8);
212 11 111 16 IF I<>0 THEN
213 11 112 23 TERMINATE
214 11 111 23 ELSE
215 11 112 28 BEGIN
216 11 113 28 FOR I:=1 TO NMEASURES DO
217 11 114 44 BEGIN
218 11 115 44 GET(MEASFILE);
219 11 115 52 MEASCORE[I]:=MEASFILE^;
220 11 114 80 END;
221 11 113 90 GET(MEASFILE);
222 11 113 98 MEASLAST:=MEASFILE^;
223 11 113 114 NMEASLAST:=TRUNC(MEASLAST);
224 11 113 127 CLOSE(MEASFILE);
225 11 112 136 END;
226 11 110 136 END;
227 11 110 150

```

READMEASFILE loads core with index to measures file.

```

228 12 11D 1 (86P8)SEGMENT PROCEDURE OPENDATAFILE;
229 12 110 0 BEGIN
230 12 110 0 (86I-8)
231 12 111 0 RESET(DATANODE,DATANAME);
232 12 111 11 (86I+8)
233 12 111 11 I:=IORESULT;
234 12 111 16 IF I<>0 THEN
235 12 112 23 BEGIN
236 12 113 23 WRITE('DATABASE MUST BE CREATED BEFORE IT CAN BE PRINTED',CHR(13),
237 12 113 94 '... ALSO');
238 12 113 114 TERMINATE;
239 12 112 117 END;
240 12 111 117 CLOSE(DATANODE);
241 12 110 126 END;
242 12 110 138

```

OPENDATAFILE checks to be sure performance item file exists.

```

243 13 110 1 (SSP)SEGMENT PROCEDURE DEFINEASPECTS;
244 13 110 0 BEGIN
245 13 111 0 ASPECT[1]:='Potentialities';
246 13 111 30 ASPECT[2]:='Processes';
247 13 111 55 ASPECT[3]:='Products';
248 13 111 79 ASPECT[4]:='Environment';
249 13 111 104 ASPECT[5]:='Constraints';
250 13 110 133 END;
251 13 110 144

```

DEFINEASPECTS tells the computer the labels for the aspects.

```

252 14 110 1 (80P)SEGMENT PROCEDURE READCOREFILE;
253 14 110 0 BEGIN
254 14 110 0 (80I-8)
255 14 111 0 RESET(COREFILE,CORENAME);
256 14 111 11 I:=IORESULT;
257 14 111 16 (80I+8)
258 14 111 16 IF I<>0 THEN
259 14 112 23 BEGIN
260 14 113 23 IF I=9 THEN
261 14 114 30 BEGIN
262 14 115 30 PAGE(OUTPUT);
263 14 115 40 WRITELN('THE APMDISK IS NOT MOUNTED');
264 14 115 86 WRITELN('');
265 14 115 106 WRITELN('PLEASE PLACE IT IN DRIVE 02');
266 14 115 153 ANYKEY;
267 14 115 156 READCOREFILE;
268 14 115 158 EXIT(READCOREFILE)
269 14 114 162 END
270 14 113 162 ELSE
271 14 114 164 BEGIN
272 14 115 164 WRITELN('COREFILE DOES NOT EXIST');
273 14 115 207 WRITELN(' ***FATAL ERROR*** ');
274 14 115 250 WRITELN(' ',I);
275 14 115 293 ANYKEY;
276 14 115 294 BRANCHOUT;
277 14 115 299 SETCHAIN('GREETING');
278 14 115 313 EXIT(PROGRAM);
279 14 114 317 END;
280 14 112 317 END
281 14 111 317 ELSE
282 14 112 319 FOR I:=1 TO 300 DO
283 14 113 335 BEGIN
284 14 114 335 GET(COREFILE);
285 14 114 343 CORE[I]:=COREFILE^;
286 14 113 371 END;
287 14 111 381 GET(COREFILE);
288 14 111 389 CORELAST:=COREFILE^;
289 14 111 405 NCORELAST:=TRUNC(CORELAST);
290 14 111 418 CLOSE(COREFILE)
291 14 110 427 END;
292 14 110 448

```

READCOREFILE reads index to performance items into core.

```

293 15 11D 1 (84P8)SEGMENT PROCEDURE SORTATTRFILE;
294 15 110 0 BEGIN
295 15 111 0 IF NATRLAST<2 THEN
296 15 112 7 EXIT(SORTATTRFILE);
297 15 111 11 FOR I:=1 TO NATRIBUTES DO
298 15 112 27 ATTR2[I]:=I;
299 15 111 56 IF NATRLAST<2 THEN
300 15 112 63 EXIT(SORTATTRFILE);
301 15 111 67 I:=2;
302 15 111 71 REPEAT
303 15 112 71 IF ATTRCORE[I]<ATTRCORE[I-1] THEN
304 15 113 116 BEGIN
305 15 114 116 TEMPX:=ATTRCORE[I];
306 15 114 144 ATTRCORE[I]:=ATTRCORE[I-1];
307 15 114 186 ATTRCORE[I-1]:=TEMPX;
308 15 114 216 TEMP2:=ATTR2[I];
309 15 114 235 ATTR2[I]:=ATTR2[I-1];
310 15 114 249 ATTR2[I-1]:=TEMP2;
311 15 114 290 IF I>2 THEN
312 15 115 297 I:=I-1;
313 15 113 305 END
314 15 112 305 ELSE
315 15 113 307 I:=I+1;
316 15 111 315 UNTIL I>NATRLAST;
317 15 110 324 END;
318 15 110 342

```

SORTATTRFILE forms an array ATTR2 which is a permutation vector for the attributes file so that if one were to print out ATTRIBUTES [ATTR2(I)] for I=1 to NATRIBUTES, the attributes would appear in numerical order.



```

319 16 110 1 ($0P)SEGMENT PROCEDURE SORTMEASFILE;
320 16 110 0 BEGIN
321 16 111 0 IF NMEASLAST<2 THEN
322 16 112 7 EXIT(SORTMEASFILE);
323 16 111 11 FOR I:=1 TO NMEASURES DO
324 16 112 27 MEAS2[I]:=I;
325 16 111 56 IF NMEASLAST<2 THEN
326 16 112 63 EXIT(SORTMEASFILE);
327 16 111 67 I:=2;
328 16 111 71 REPEAT
329 16 112 71 IF MEASCORE[I]<MEASCORE[I-1] THEN
330 16 113 116 BEGIN
331 16 114 116 TEMPX:=MEASCORE[I];
332 16 114 144 MEASCORE[I]:=MEASCORE[I-1];
333 16 114 186 MEASCORE[I-1]:=TEMPX;
334 16 114 216 TEMP2:=MEAS2[I];
335 16 114 235 MEAS2[I]:=MEAS2[I-1];
336 16 114 269 MEAS2[I-1]:=TEMP2;
337 16 114 290 IF I>2 THEN
338 16 115 297 I:=I-1;
339 16 113 305 END
340 16 112 305 ELSE
341 16 113 307 I:=I+1;
342 16 111 315 UNTIL I>NMEASLAST;
343 16 110 324 END;
344 16 110 342

```

SORTMEASFILE forms an array MEAS2 which is a permutation vector for the measures file.

```

345 17 110 1 ($9P$)SEGMENT PROCEDURE SORTCOREFILE;
346 17 110 0 BEGIN
347 17 111 0 FOR I:=1 TO 300 DO
348 17 112 16 CORE2[I]:=I;
349 17 111 45 IF NCORELAST<2 THEN
350 17 112 52 EXIT(SORTCOREFILE);
351 17 111 56 I:=2;
352 17 111 60 REPEAT
353 17 112 60 IF CORE[I]<CORE[I-1] THEN
354 17 113 105 BEGIN
355 17 114 105 TEMP:=CORE[I];
356 17 114 133 CORE[I]:=CORE[I-1];
357 17 114 175 CORE[I-1]:=TEMP;
358 17 114 205 TEMP2:=CORE2[I];
359 17 114 224 CORE2[I]:=CORE2[I-1];
360 17 114 258 CORE2[I-1]:=TEMP2;
361 17 114 279 IF I>2 THEN
362 17 115 286 I:=I-1;
363 17 113 294 END
364 17 112 294 ELSE
365 17 113 296 I:=I+1;
366 17 111 304 UNTIL I>NCORELAST;
367 17 110 313 END;
368 17 110 332

```

SORTCOREFILE prepares an array CORE2 which lists the location of each performance item in numeric order.

```

3691811D 1($P$)SEGMENT PROCEDURE NAMEFILES;
37018110 0 BEGIN
37118111 0 APHDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
37218111 86 NAMEATCORE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'AC');
37318111182 NAMEATTRIBUTES:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,
1,4)),'AT');
37418111278 NAMEMEASURE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'MC');
37518111374 NAMEMEASURES:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,
1,4)),'ME');
37618111470 CORENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
37718111566 DATANAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
37818111662 ISSUENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
37918111758 NAMEFASTISSUE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FA');
38018111854 END;
38118111866

```

NAMEFILES constructs strings containing the names of files used in this program.

```

382 19 110 1 (80P)SEGMENT PROCEDURE DISPLAYNAME;
383 19 110 0 BEGIN
384 19 111 0 SEEK(ISSUE,I);
385 19 111 11 GET(ISSUE);
386 19 111 19 WRITE(I,'. ');
387 19 111 44 FOR J:=1 TO 2 DO
388 19 112 60 BEGIN
389 19 113 60 IF LENGTH(ISSUE^.NAME[J])>60 THEN
390 19 114 81 LINE:=COPY(ISSUE^.NAME[J],1,60)
391 19 113 108 ELSE
392 19 114 112 LINE:=ISSUE^.NAME[J];
393 19 113 132 IF J=2 THEN
394 19 114 139 WRITE(' ');
395 19 113 153 WRITELN(LINE);
396 19 112 173 END;
397 19 111 183 WRITELN(' ');
398 19 110 201 END;
399 19 110 216

```

DISPLAYNAME displays a measurement purpose on screen [called by display issues].

```

400 20 1:D 1($P$)SEGMENT PROCEDURE DISPLAYISSUES;
401 20 1:0 0 BEGIN
402 20 1:0 0 ($I-$)
403 20 1:1 0 RESET(ISSUE,ISSUENAME);
404 20 1:1 11 ($I+$)
405 20 1:1 11 PAGE(OUTPUT);
406 20 1:1 21 IF IORESULT<>0 THEN
407 20 1:2 27 BEGIN
408 20 1:3 27 NUISSUES:=0;
409 20 1:3 31 EXIT(DISPLAYISSUES);
410 20 1:2 35 END;
411 20 1:1 35 IF NUISSUES=0 THEN
412 20 1:2 42 BEGIN
413 20 1:3 42 Writeln('Currently, there are no measurement issues in the APM for this
system and subsystem');
414 20 1:2 145 END
415 20 1:1 145 ELSE
416 20 1:2 147 BEGIN
417 20 1:3 147 Writeln('The following measurement purposes are currently included in the APM:');
418 20 1:3 237 FOR I:=1 TO NUISSUES DO
419 20 1:4 253 BEGIN
420 20 1:5 253 DISPLAYNAME;
421 20 1:5 254 IF (I MOD 6=0) THEN
422 20 1:6 265 BEGIN
423 20 1:7 265 ANYKEY;
424 20 1:7 268 PAGE(OUTPUT);
425 20 1:6 278 END;
426 20 1:4 278 END;
427 20 1:2 288 END;
428 20 1:1 288 CLOSE(ISSUE);
429 20 1:0 297 END;
430 20 1:0 314
430 20 1:0 314($I $S:UTILITY.TEXT$)
431 20 1:0 314

```

DISPLAYISSUES displays names of all measurement purposes on screen.

```

432 1 2:0 1 ($P$)PROCEDURE ANYKEY;
433 1 2:0 0 BEGIN
434 1 2:1 0 WRITELN(' ');
435 1 2:1 16 WRITELN('$$$ Please press any key to continue $$$');
436 1 2:1 78 ($R-8)
437 1 2:1 78 READ(ANS);
438 1 2:1 89 ($R+8)
439 1 2:0 89 END;
440 1 2:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

441 1 610 1 (80PS)PROCEDURE HELPER;
442 1 610 0 BEGIN
443 1 611 0 WRITELN('For help please refer to your APN MANUAL.');
```

```

444 1 610 61 END;
445 1 610 74
```

HELPER due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```

446 1 7:0 1 (88P)PROCEDURE KEYN;
447 1 7:0 1 VAR
448 1 7:0 1 ANSWER: STRING[40];
449 1 7:0 22 I1: ARRAY[1..4] OF INTEGER;
450 1 7:0 26 OK: BOOLEAN;
451 1 7:0 27 I10: INTEGER;
452 1 7:0 28
453 1 7:0 0 BEGIN
454 1 7:0 0 (88R-8)
455 1 7:1 0 REPEAT
456 1 7:2 0 REPEAT
457 1 7:3 0 ANSWER:=
458 1 7:3 27 OK:=TRUE;
459 1 7:3 30 READLN(ANSWER);
460 1 7:3 49 IF LENGTH(ANSWER)=0 THEN
461 1 7:4 57 Writeln('Please enter the integer again');
462 1 7:2 107 UNTIL LENGTH(ANSWER)<>0;
463 1 7:2 115 IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
464 1 7:3 130 HELPER;
465 1 7:2 132 FOR I:=1 TO 4 DO
466 1 7:3 147 BEGIN
467 1 7:4 147 I1[I]:=ORD(ANSWER[I])-48;
468 1 7:4 165 IF (I1[I]<0) OR (I1[I]>9) THEN
469 1 7:5 192 BEGIN
470 1 7:6 192 IF (I=1) OR (I1[I]<>(ORD(' ')-48)) THEN
471 1 7:7 214 BEGIN
472 1 7:8 214 OK:=FALSE;
473 1 7:8 217 Writeln('PLEASE RESPOND WITH A POSITIVE INTEGER');
474 1 7:7 275 END;
475 1 7:5 275 END;
476 1 7:3 275 END;
477 1 7:1 285 UNTIL OK=TRUE;
478 1 7:1 292 I10:=I1[1];
479 1 7:1 302 FOR I:=2 TO 4 DO
480 1 7:2 317 BEGIN
481 1 7:3 317 IF (I1[I]=0) AND (I1[I]<=9) THEN
482 1 7:4 344 I10:=I10*10+I1[I];
483 1 7:2 361 END;
484 1 7:2 371 (88R+8)
485 1 7:1 371 I:=I10;
486 1 7:0 376 END;
487 1 7:0 398

```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.



```

488 1 0:0 1 (80R)PROCEDURE KEY;
489 1 0:0 1 VAR
490 1 0:0 1 II2:INTEGER;
491 1 0:0 0 BEGIN
492 1 0:0 0 (80R-8)
493 1 0:1 0 ANSWER:=
494 1 0:1 27 REPEAT
495 1 0:2 27 READLN(ANSWER);
496 1 0:2 47 ANSI=ANSWER[1];
497 1 0:2 55 IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
498 1 0:2 78 (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27) THEN
499 1 0:3 98 WRITELN('PLEASE RESPOND YES OR NO!');
500 1 0:2 143 IF (ORD(ANS)>90) THEN
501 1 0:3 150 BEGIN
502 1 0:4 150 II2:=ORD(ANS)-32;
503 1 0:4 157 ANSI=CHR(II2);
504 1 0:3 161 END;
505 1 0:1 161 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
506 1 0:1 186 (80R+8)
507 1 0:1 186 IF ANS='H' THEN
508 1 0:2 193 HELPER;
509 1 0:0 195 END;
510 1 0:0 210

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

S11 1 910 1 (80P)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
S12 1 910 0 BEGIN
S13 1 911 0 HLP:=HLP;
S14 1 911 9 REPEAT
S15 1 912 9 WRITE(MSG);
S16 1 912 20 KEY;
S17 1 911 22 UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
S18 1 910 41 END;
S19 1 910 56

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

520 1 10:0 1 (80P)PROCEDURE INLINE;
521 1 10:0 1 VAR
522 1 10:0 1 LONGLINE:STRING[125];
523 1 10:0 64 LINEOK:BOOLEAN;
524 1 10:0 65
525 1 10:0 0 BEGIN
526 1 10:1 0 REPEAT
527 1 10:2 0 READLN(LONGLINE);
528 1 10:2 19 LINEOK:=TRUE;
529 1 10:2 22 M:=LENGTH(LONGLINE);
530 1 10:2 29 IF M>80 THEN
531 1 10:3 36 BEGIN
532 1 10:4 36 WRITELN('WARNING LINE CONTAINS OVER 80 CHARACTERS');
533 1 10:4 100 WRITELN(' ');
534 1 10:4 118 PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 80 CHARACTERS? ');
535 1 10:4 166 IF ANS='N' THEN
536 1 10:5 173 BEGIN
537 1 10:6 173 LINEOK:=FALSE;
538 1 10:6 176 WRITELN('PLEASE TYPE LINE AGAIN: ');
539 1 10:5 220 END
540 1 10:4 220 ELSE
541 1 10:5 222 M:=80;
542 1 10:3 226 END;
543 1 10:1 226 UNTIL LINEOK;
544 1 10:1 230 LINER:=COPY(LONGLINE,1,M);
545 1 10:0 248 END;
546 1 10:0 264

```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```

547 1 3:0 1 (800)PROCEDURE BRANCHIN;
548 1 3:0 0 BEGIN
549 1 3:0 0 (801-8)
550 1 3:1 0 RESET(PASSNODE,'PASSTHRU');
551 1 3:1 18 I:=IORESULT;
552 1 3:1 23 (801+8)
553 1 3:1 23 IF I<>0 THEN
554 1 3:2 30 BEGIN
555 1 3:3 30 WRITELN('PASSTHRU FILE DOES NOT EXIST');
556 1 3:3 78 WRITELN(' *****FATAL ERROR*****');
557 1 3:3 123 WRITELN(' ',1);
558 1 3:3 147 ANYKEY;
559 1 3:3 169 SETCHAIN('PGH1');
560 1 3:3 179 EXIT(PROGRAM);
561 1 3:2 183 END;
562 1 3:1 183 GET(PASSNODE);
563 1 3:1 190 CURSYS:=PASSNODE^.CURSYS;
564 1 3:1 198 CURSP:=PASSNODE^.CURSP;
565 1 3:1 206 CURSUB:=PASSNODE^.CURSUB;
566 1 3:1 214 PAC:=PASSNODE^.PAC;
567 1 3:1 220 NCURSYS:=PASSNODE^.NCURSYS;
568 1 3:1 227 NCURSP:=PASSNODE^.NCURSP;
569 1 3:1 234 NCURSUB:=PASSNODE^.NCURSUB;
570 1 3:1 241 NPAC:=PASSNODE^.NPAC;
571 1 3:1 248 CLOSE(PASSNODE);
572 1 3:0 256 END;
573 1 3:0 270

```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```

574 1 4:D 1 ($P$)PROCEDURE BRANCHOUT;
575 1 4:0 0 BEGIN
576 1 4:1 0 REWRITE(PASSNODE,'PASSTHRU');
577 1 4:1 20 PASSNODE^.FLAG1:=1;
578 1 4:1 26 PUT(PASSNODE);
579 1 4:1 33 CLOSE(PASSNODE,LOCK);
580 1 4:0 41 END;
581 1 4:0 54
582 1 4:0 54
583 1 4:0 54
584 1 4:0 54 ($I $5:UTILITY.TEXT$)
585 1 4:0 54
585 1 4:0 54 ($I $5:PRINT2.TEXT$)

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```

586 1 11:0 1 (80P)PROCEDURE TOPSCREEN;
587 1 11:0 0 BEGIN
588 1 11:1 0 PAGE(OUTPUT);
589 1 11:1 10 M:=LENGTH(CURSYS);
590 1 11:1 18 IF M>16 THEN
591 1 11:2 25 M:=16;
592 1 11:1 29 LINE:=COPY(CURSYS,1,M);
593 1 11:1 48 WRITE('S',LINE,' Systems');
594 1 11:1 90 GOTOXY(26,0);
595 1 11:1 95 M:=LENGTH(CURSP);
596 1 11:1 103 IF M>16 THEN
597 1 11:2 110 M:=16;
598 1 11:1 114 LINE:=COPY(CURSP,1,M);
599 1 11:1 133 WRITE('S',LINE);
600 1 11:1 155 GOTOXY(44,0);
601 1 11:1 160 M:=LENGTH(CURSUB);
602 1 11:1 168 IF M>16 THEN
603 1 11:2 175 M:=16;
604 1 11:1 179 LINE:=COPY(CURSUB,1,M);
605 1 11:1 198 WRITELN('S',LINE);
606 1 11:1 228 GOTOXY(62,0);
607 1 11:1 233 WRITELN('S',PAC);
608 1 11:1 263 M:=LENGTH(XOBJECTIVE);
609 1 11:1 271 IF M>67 THEN M:=67;
610 1 11:1 282 LINE:=COPY(XOBJECTIVE,1,M);
611 1 11:1 301 IF NSCREEN>1 THEN
612 1 11:2 308 WRITELN('Objective['',MOBJECTIVE,'']',LINE);
613 1 11:1 376 M:=LENGTH(XFUNPUR);
614 1 11:1 384 IF M>67 THEN M:=67;
615 1 11:1 395 LINE:=COPY(XFUNPUR,1,M);
616 1 11:1 414 IF NSCREEN>2 THEN
617 1 11:2 421 WRITELN('Fct1 Prps['',NFUNPUR,'']',LINE);
618 1 11:1 489 WRITELN(' ');
619 1 11:0 507 END;
620 1 11:0 520

```

TOPSCREEN displays appropriate header information at the top of each screen.

```

621 1 12:0 1 ($P$)PROCEDURE WHICHELIMINATE;
622 1 12:0 0 BEGIN
623 1 12:1 0 PAGE(OUTPUT);
624 1 12:1 10 WRITELN('Would you like to eliminate at the level of:',chr(13),
625 1 12:1 76 ' 0. No elimination',chr(13),
626 1 12:1 117 ' 1. Objectives',chr(13),
627 1 12:1 154 ' 2. Functional Purposes',chr(13),
628 1 12:1 200 ' 3. Characteristics');
629 1 12:1 240 REPEAT
630 1 12:2 240 KEYN;
631 1 12:2 242 IF (I<0) OR (I>3) OR (I<0) THEN
632 1 12:3 261 WRITELN('PLEASE SPECIFY AN INTEGER BETWEEN 0 AND 3');
633 1 12:1 322 UNTIL (I>=0) OR (I<4);
634 1 12:1 335 IF I=0 THEN
635 1 12:2 342 BEGIN
636 1 12:3 342 CLOSE(DATANODE);
637 1 12:3 351 EXIT(ELIMINATE);
638 1 12:2 355 END;
639 1 12:1 355 IF I=1 THEN
640 1 12:2 362 BEGIN
641 1 12:3 362 LEVEL:='OBJ';
642 1 12:3 373 CUT:=10000;
643 1 12:3 379 NSCREEN:=1;
644 1 12:2 383 END;
645 1 12:1 383 IF I=2 THEN
646 1 12:2 390 BEGIN
647 1 12:3 390 LEVEL:='FP';
648 1 12:3 400 CUT:=100;
649 1 12:3 404 NSCREEN:=2;
650 1 12:2 408 END;
651 1 12:1 408 IF I=3 THEN
652 1 12:2 415 BEGIN
653 1 12:3 415 LEVEL:='CHAR';
654 1 12:3 427 CUT:=1;
655 1 12:3 431 NSCREEN:=3;
656 1 12:2 435 END;
657 1 12:0 435 END;
658 1 12:0 450

```

WHICHELIMINATE asks what level should be used in asking analyst what performance items are not part of his/her analysis.

```

659 1 13:D 1 (26P)PROCEDURE ASKELIMINATE;
660 1 13:0 0 BEGIN
661 1 13:1 0 J:=TRUNC(CORE[NODE] DIV 1000000);
662 1 13:1 54 PAC:=ASPECT[J];
663 1 13:1 72 NPAC:=J;
664 1 13:1 78 TOPSCREEN;
665 1 13:1 80 WRITELN('The following taxon is scheduled to be printed: ');
666 1 13:1 148 SEEK(DATANODE,CORE2[NODE]);
667 1 13:1 172 GET(DATANODE);
668 1 13:1 180 GOTOXY(0,12);
669 1 13:1 185 WRITE(CHR(11),' ');
670 1 13:1 209 FOR J:=1 TO 4 DO
671 1 13:2 223 WRITE(DATANODE^.NTAXA[J],'.');
672 1 13:1 266 WRITELN(DATANODE^.TAXA);
673 1 13:1 288 WRITELN(' ');
674 1 13:1 306 PREPKEY(230,'Would you like to print it?');
675 1 13:1 341 IF ORD(ANS)=27 THEN
676 1 13:2 348 BEGIN
677 1 13:3 348 CLOSE(DATANODE);
678 1 13:3 357 EXIT(ELIMINATE);
679 1 13:2 361 END;
680 1 13:1 361 IF ANS='Y' THEN
681 1 13:2 368 EXIT(ASKELIMINATE);
682 1 13:1 372 PRINTIT[NODE]:=FALSE;
683 1 13:1 390 IF LEVEL<>'CHAR' THEN
684 1 13:2 404 FOR I:=1 TO NCORELAST DO
685 1 13:3 420 IF CORE[I] DIV CUT * CUT = CORE[NODE]
686 1 13:3 475 THEN PRINTIT[I]:=FALSE;
687 1 13:0 513 END;
688 1 13:0 530

```

ASKELIMINATE asks analyst exactly what he/she wants to eliminate by presenting taxons one at a time.



```

689 1 5:0 1 (##P#)PROCEDURE ELIMINATE;
690 1 5:0 0 BEGIN
691 1 5:1 0 PREPKEY(260,'Do you wish to eliminate any performance items from your printout?');
692 1 5:1 74 IF (ANS='N') OR (ORD(ANS)=27) THEN
693 1 5:2 87 EXIT(ELIMINATE);
694 1 5:1 91 RESET(DATANODE,DATANAME);
695 1 5:1 104 WHICHELIMINATE;
696 1 5:1 106 FOR NODE:=1 TO NCORELAST DO
697 1 5:2 122 IF CORE[NODE]<>0 THEN
698 1 5:3 152 BEGIN
699 1 5:4 152 IF LEVEL='OBJ' THEN
700 1 5:5 165 IF PRINTIT[NODE]=TRUE THEN
701 1 5:6 187 IF CORE[NODE]=CORE[NODE] DIV 10000 * 10000 THEN
702 1 5:7 252 ASKELIMINATE;
703 1 5:4 254 IF LEVEL='FP' THEN
704 1 5:5 266 IF PRINTIT[NODE]=TRUE THEN
705 1 5:6 288 IF CORE[NODE]=CORE[NODE] DIV 100 * 100 THEN
706 1 5:7 349 IF CORE[NODE]<>CORE[NODE] DIV 10000 * 10000 THEN
707 1 5:8 414 ASKELIMINATE;
708 1 5:4 416 IF LEVEL='CHAR' THEN
709 1 5:5 430 IF PRINTIT[NODE]=TRUE THEN
710 1 5:6 452 IF CORE[NODE]<>CORE[NODE] DIV 100 * 100 THEN
711 1 5:7 513 ASKELIMINATE;
712 1 5:4 515 IF NSCREEN>1 THEN
713 1 5:5 522 IF CORE[NODE] DIV 10000 * 10000=CORE[NODE] THEN
714 1 5:6 587 BEGIN
715 1 5:7 587 SEEK(DATANODE,CORE2[NODE]);
716 1 5:7 611 GET(DATANODE);
717 1 5:7 619 XOBJECTIVE:=DATANODE^.TAXA;
718 1 5:7 629 NOBJECTIVE:=DATANODE^.NTAXAC2;
719 1 5:6 644 END;
720 1 5:4 644 IF NSCREEN>2 THEN
721 1 5:5 651 IF CORE[NODE] DIV 100 * 100 = CORE[NODE] THEN
722 1 5:6 712 BEGIN
723 1 5:7 712 SEEK(DATANODE,CORE2[NODE]);
724 1 5:7 736 GET(DATANODE);
725 1 5:7 744 XFUNPUR:=DATANODE^.TAXA;
726 1 5:7 754 NFUNPUR:=DATANODE^.NTAXAC3;
727 1 5:6 769 END;
728 1 5:3 769 END;
729 1 5:1 779 CLOSE(DATANODE);
730 1 5:0 788 END;
731 1 5:0 810

```

ELIMINATE using information gained from WHICHELIMINATE, calls ASKELIMINATE as appropriate.

```

732 1 14:0 1  (%%P*)PROCEDURE PROCESSISSUE;
733 1 14:0 0  BEGIN
734 1 14:1 0  IF NOISSUE=FALSE THEN
735 1 14:2 8  BEGIN
736 1 14:3 8  REPEAT
737 1 14:4 8  WRITE('Which measurement purpose would you like to use (type 0 for none)?');
738 1 14:4 84  KEYN;
739 1 14:3 88  UNTIL (I>=0) AND (I<=NISSUES)
740 1 14:2 100  END
741 1 14:1 103  ELSE
742 1 14:2 105  I:=0;
743 1 14:1 109  NCURISSUE:=I;
744 1 14:1 115  IF I=0 THEN
745 1 14:2 122  BEGIN
746 1 14:3 122  FOR J:=1 TO 300 DO
747 1 14:4 138  PRINTIT(J):=TRUE;
748 1 14:3 166  EXIT (PROCESSISSUE);
749 1 14:2 170  END;
750 1 14:1 170  RESET(FASTISSUE,NAMEFASTISSUE);
751 1 14:1 183  SEEK(FASTISSUE,I);
752 1 14:1 194  GET(FASTISSUE);
753 1 14:1 202  CLOSE(FASTISSUE);
754 1 14:1 211  OK:=FALSE;
755 1 14:1 215  FOR J:=1 TO 300 DO
756 1 14:2 231  BEGIN
757 1 14:3 231  PRINTIT(J):=FASTISSUE^.PRINTIT(J);
758 1 14:3 265  IF PRINTIT(J)=TRUE THEN
759 1 14:4 287  OK:=TRUE;
760 1 14:2 291  END;
761 1 14:1 301  IF OK=TRUE THEN
762 1 14:2 309  EXIT(PROCESSISSUE);
763 1 14:1 313  FOR J:=1 TO 300 DO
764 1 14:2 329  PRINTIT(J):=FALSE;
765 1 14:1 357  RESET(ISSUE,ISSUENAME);
766 1 14:1 370  SEEK(ISSUE,I);
767 1 14:1 381  GET(ISSUE);
768 1 14:1 389  T2:=0;
769 1 14:1 404  Writeln('Please be patient...',chr(13),
770 1 14:1 446  ' I am getting set up to use your measurement purpose');
771 1 14:1 519  FOR J:=1 TO 225 DO

```

PROCESSISSUE selects performance items for printing based upon the measurement purpose in use.

772	1	14:2	535	BEGIN
773	1	14:3	535	T1:=ISSUE^.DATA[J];
774	1	14:3	565	IF T1<>0 THEN
775	1	14:4	583	BEGIN
776	1	14:5	583	SKIP:=FALSE;
777	1	14:5	587	FOR K:=1 TO 300 DO
778	1	14:6	603	BEGIN
779	1	14:7	603	T5:=CORE[K];
780	1	14:7	631	IF (T1 = T5) THEN
781	1	14:8	650	BEGIN
782	1	14:9	650	PRINTIT[K]:=TRUE;
783	1	14:9	668	SKIP:=TRUE;
784	1	14:8	672	END;
785	1	14:6	672	END;
786	1	14:5	682	IF SKIP=TRUE THEN
787	1	14:6	690	IF (T1 DIV 100 * 100 <> T2) THEN
788	1	14:7	727	BEGIN
789	1	14:8	727	FOR K:=1 TO 300 DO
790	1	14:9	743	BEGIN
791	1	14:0	743	T2:=T1 DIV 100*100;
792	1	14:0	777	T3:=T1 DIV 10000 * 10000;
793	1	14:0	815	T4:=T1 DIV 1000000 \$1000000;
794	1	14:0	889	T5:=CORE[K];
795	1	14:0	917	IF T2 = T5 THEN
796	1	14:1	936	PRINTIT[K]:=TRUE;
797	1	14:0	954	IF T3 = T5 THEN
798	1	14:1	973	PRINTIT[K]:=TRUE;
799	1	14:0	991	IF T4 = T5 THEN
800	1	14:1	1010	PRINTIT[K]:=TRUE;
801	1	14:9	1028	END;
802	1	14:7	1038	END;
803	1	14:4	1038	END;
804	1	14:2	1038	END;
805	1	14:1	1048	CLOSE(ISSUE);
806	1	14:1	1057	RESET(FASTISSUE,NAMEFASTISSUE);
807	1	14:1	1070	J:=NCURISSUE;
808	1	14:1	1076	SEEK(FASTISSUE,J);
809	1	14:1	1087	FOR J:=1 TO 300 DO
810	1	14:2	1103	FASTISSUE^.PRINTIT[J]:=PRINTIT[J];
811	1	14:1	1147	PUT(FASTISSUE);
812	1	14:1	1155	CLOSE(FASTISSUE);
813	1	14:0	1164	END;
814	1	14:0	1202	

See previous page for program description.

```

015 1 15:0 1 (80P)PROCEDURE PRINTMEASURE;
016 1 15:0 0 BEGIN
017 1 15:1 0 FOR NCURMEASURE:=1 TO NMEASLAST DO
018 1 15:2 16 IF ATTRCORE[NCURATTRIBUTE]=MEASCORE[NCURMEASURE] DIV 100 THEN
019 1 15:3 68 BEGIN
020 1 15:4 68 SEEK(MEASURES,MEAS2[NCURMEASURE]);
021 1 15:4 92 GET(MEASURES);
022 1 15:4 100 WRITE(PRNT,' ');
023 1 15:4 124 FOR K:=1 TO 6 DO
024 1 15:5 138 WRITE(PRNT,MEASURES^.NDESCRIPTOR[K],'.');
025 1 15:4 181 WRITELN(PRNT,' ',MEASURES^.DESCRIPTOR);
026 1 15:3 213 END;
027 1 15:0 223 END;
028 1 15:0 244

```

PRINTMEASURE prints a measure for current performance item.

```

829 1 161D 1 (84P8)PROCEDURE PRINTATTRIBUTE;
830 1 1610 0 BEGIN
831 1 1611 0 FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
832 1 1612 16 IF CORE[1]=ATTRCORE[NCURATTRIBUTE] DIV 100 THEN
833 1 1613 68 BEGIN
834 1 1614 68 SEEK(ATTRIBUTES,ATTR2[NCURATTRIBUTE]);
835 1 1614 92 GET(ATTRIBUTES);
836 1 1614 100 WRITE(PRNT,' ');
837 1 1614 122 FOR K:=1 TO 6 DO
838 1 1615 136 WRITE(PRNT,ATTRIBUTES^.NDESCRIPTOR(K),'.');
839 1 1614 179 WRITELN(PRNT,' ',ATTRIBUTES^.DESCRIPTOR);
840 1 1614 211 PRINTMEASURE;
841 1 1613 213 END;
842 1 1610 223 END;
843 1 1610 244

```

PRINTATTRIBUTE prints attributes for current performance items.

```

844 1 17:0 1($Ps)PROCEDURE GETUSERSTUFF;
845 1 17:0 0 BEGIN
846 1 17:1 0 USERNAME:='';
847 1 17:1 10 USERMSG:='';
848 1 17:1 34 REPEAT
849 1 17:2 34 WRITE('What is your name? ');
850 1 17:2 67 ($R-8) .
851 1 17:2 67 READLN(USERNAME);
852 1 17:2 87 ($R+8)
853 1 17:2 87 WRITELN(' ');
854 1 17:1 105 UNTIL (COPY(USERNAME,1,3) <> ' ') AND (LENGTH(USERNAME)>0);
855 1 17:1 135 REPEAT
856 1 17:2 135 WRITELN('Please type a 40-character (max.) identification code for the printout!');
857 1 17:2 226 ($R-8)
858 1 17:2 226 READLN(USERMSG);
859 1 17:2 246 ($R+8)
860 1 17:2 246 WRITELN(' ');
861 1 17:1 264 UNTIL (COPY(USERMSG,1,3) <> ' ') AND (LENGTH(USERMSG)>0);
862 1 17:1 294 REPEAT
863 1 17:2 294 WRITELN('Please type todays date: ');
864 1 17:2 339 ($R-8)
865 1 17:2 339 READLN(USERDATE);
866 1 17:2 359 ($R+8)
867 1 17:2 359 WRITELN(' ');
868 1 17:1 377 UNTIL (COPY(USERMSG,1,3) <> ' ') AND (LENGTH(USERDATE)>0);
869 1 17:0 407 END;
870 1 17:0 426

```

GETUSERSTUFF asks analyst for his/her name, project title and the date.

```

071 1 10:0 1 (000)PROCEDURE TITLEPAGE;
072 1 10:0 0 BEGIN
073 1 10:1 0 REWRITE(PRINT,'PRINTER:');
074 1 10:1 21 FOR I:=1 TO 13 DO
075 1 10:2 35 WRITELN(PRINT,CHR(14),' ');
076 1 10:1 73 WRITELN(PRINT,CHR(14),' An Analytic Process Model For');
077 1 10:1 147 WRITELN(PRINT,CHR(14),' ');
078 1 10:1 175 WRITELN(PRINT,CHR(14),' Systems Design And Measurement:');
079 1 10:1 250 FOR I:=1 TO 5 DO
080 1 10:2 264 WRITELN(PRINT,CHR(14),' ');
081 1 10:1 302 WRITELN(PRINT,CHR(14),' Listing Of Taxa and Measurements');
082 1 10:1 378 FOR I:=1 TO 5 DO
083 1 10:2 392 WRITELN(PRINT,CHR(14),' ');
084 1 10:1 430 WRITELN(PRINT,CHR(14),'For: ',USERNAME);
085 1 10:1 477 WRITELN(PRINT,CHR(14),' ');
086 1 10:1 505 WRITELN(PRINT,CHR(14),'Date: ',USERDATE);
087 1 10:1 553 WRITELN(PRINT,CHR(14),' ');
088 1 10:1 581 WRITELN(PRINT,CHR(14),'Re: ',USERMSG);
089 1 10:1 627 CLOSE(PRINT);
090 1 10:0 636 END;
091 1 10:0 654

```

TITLEPAGE prints title page for printout.

```

892 1 19:0 1 (80P8)PROCEDURE HEADER;
893 1 19:0 0 BEGIN
894 1 19:1 0 REWRITE(PRNT,'PRINTER:');
895 1 19:1 21 PAGE(PRNT);
896 1 19:1 31 WRITELN(PRNT,CHR(14),USERNAME);
897 1 19:1 41 WRITELN(PRNT,CHR(14),USERDATE);
898 1 19:1 91 WRITELN(PRNT,CHR(14),USERMSG);
899 1 19:1 121 IF NCURISSUE<>0 THEN
900 1 19:2 128 BEGIN
901 1 19:3 128 RESET(ISSUE,ISSUENAME);
902 1 19:3 141 SEEK(ISSUE,NCURISSUE);
903 1 19:3 152 GET(ISSUE);
904 1 19:3 160 WRITELN(PRNT,' ');
905 1 19:3 178 WRITELN(PRNT,CHR(14),'Measurement Purpose: ',CHR(15),ISSUE^.NAME(1));
906 1 19:3 261 WRITELN(PRNT,' ',ISSUE^.NAME(2));
907 1 19:3 328 WRITELN(PRNT,' ');
908 1 19:3 346 CLOSE(ISSUE);
909 1 19:2 355 END
910 1 19:1 355 ELSE
911 1 19:2 357 BEGIN
912 1 19:3 357 WRITELN(PRNT,' ');
913 1 19:3 375 WRITELN(PRNT,CHR(14),'Measurement Purpose: ',CHR(15),' Global');
914 1 19:3 455 WRITELN(PRNT,' ');
915 1 19:2 473 END;
916 1 19:1 473 WRITELN(PRNT,CHR(14),'System Class: ',chr(15),CURSYS,['',NCURSYS,'']);
917 1 19:1 571 WRITELN(PRNT,CHR(14),'System: ',chr(15),CURSP,['',NCURSP,'']);
918 1 19:1 663 WRITELN(PRNT,CHR(14),'Subsystem: ',chr(15),CURSUB,['',NCURSUB,'']);
919 1 19:1 758 WRITELN(PRNT,' ',CHR(15),CHR(13));
920 1 19:0 796 END;
921 1 19:0 810

```

HEADER prints header on printout.



```

922 1 20:0 1 (86P8)PROCEDURE PRNTDATASET;
923 1 20:0 0 BEGIN
924 1 20:1 0 RESET(DATANODE,DATANAME);
925 1 20:1 13 RESET(ATTRIBUTES,NAMEATTRIBUTES);
926 1 20:1 24 RESET(MEASURES,NAMEMEASURES);
927 1 20:1 39 REPEAT
928 1 20:2 39 HEADER;
929 1 20:2 41 TEMP2:=0;
930 1 20:2 45 FOR I:=1 TO NCORELAST DO
931 1 20:3 61 IF PRINTIT[I]=TRUE THEN
932 1 20:4 83 BEGIN
933 1 20:5 83 SEEK(DATANODE,CORE2[I]);
934 1 20:5 107 GET(DATANODE);
935 1 20:5 115 INDENT:=4;
936 1 20:5 119 IF DATANODE^.NTAXA[4]=0 THEN
937 1 20:6 135 INDENT:=3;
938 1 20:5 139 IF DATANODE^.NTAXA[3]=0 THEN
939 1 20:6 155 INDENT:=2;
940 1 20:5 159 IF DATANODE^.NTAXA[2]=0 THEN
941 1 20:6 175 INDENT:=1;
942 1 20:5 179 IF (DATANODE^.NTAXA[1]>TEMP2) AND (DATANODE^.NTAXA[1]>0) THEN
943 1 20:6 212 BEGIN
944 1 20:7 212 WRITE(PRNT,DATANODE^.NTAXA[1],'.0.0.0.0. ');
945 1 20:7 257 WRITE(PRNT,ASPECT[DATANODE^.NTAXA[1]],': ');
946 1 20:7 302 CASE DATANODE^.NTAXA[1] OF
947 1 20:7 316 1:WRITELN(PRNT,OBJLBL1);
948 1 20:7 368 2:WRITELN(PRNT,OBJLBL2);
949 1 20:7 441 3:WRITELN(PRNT,OBJLBL3);
950 1 20:7 487 4:WRITELN(PRNT,OBJLBL4);
951 1 20:7 552 5:WRITELN(PRNT,OBJLBL5);
952 1 20:7 617 END;
953 1 20:6 634 END;
954 1 20:5 634 TEMP2:=DATANODE^.NTAXA[1];
955 1 20:5 649 FOR J:=1 TO INDENT DO
956 1 20:6 665 WRITE(PRNT,' ');
957 1 20:5 689 FOR J:=1 TO 4 DO
958 1 20:6 703 WRITE(PRNT,DATANODE^.NTAXA[J],'.');
959 1 20:5 746 WRITELN(PRNT,'0.0. ',DATANODE^.TAXA);
960 1 20:5 785 IF INDENT>1 THEN
961 1 20:6 792 PRINTATTRIBUTE;

```

PRNTDATASET is the controlling program for printing a data set. Also, prints all performance items. Calls PRINTATTRIBUTE when necessary.

```

962 1 2015 794      IF KEYPRESS THEN
963 1 2016 801      BEGIN
964 1 2017 801      READ(ANS);
965 1 2017 812      IF ORD(ANS)=27 THEN
966 1 2018 819      BEGIN
967 1 2019 819      WRITELN(PRNT,'Job cancelled');
968 1 2019 852      PAGE(PRNT);
969 1 2019 842      CLOSE(PRNT);
970 1 2019 871      CLOSE(DATANODE);
971 1 2019 880      CLOSE(ATTRIBUTES);
972 1 2019 889      CLOSE(MEASURES);
973 1 2019 898      EXIT(PRNTDATASET);
974 1 2018 902      END;
975 1 2016 902      END;
976 1 2014 902      END;
977 1 2012 912      PAGE(PRNT);
978 1 2012 922      CLOSE(PRNT);
979 1 2012 931      PREPKEY(303,'Would you like to print another copy of these measurements?');
980 1 2011 998      UNTIL (ANS='N') OR (ORD(ANS)=27);
981 1 2011 1011     CLOSE(DATANODE);
982 1 2011 1020     CLOSE(ATTRIBUTES);
983 1 2011 1029     CLOSE(MEASURES);
984 1 2010 1038     END;
985 1 2010 1048
986 1 2010 1048
987 1 2010 1068 (201 05:PRINT2.TEXT*)
988 1 2010 1068

```

See previous page for program description.

```

989 1 1:0 0 ($P$)BEGIN
990 1 1:0 0 ($N-$)
991 1 1:1 0 NISSUES:=5;
992 1 1:1 118 NMEASURES:=400;
993 1 1:1 124 NATTRIBUTES:=200;
994 1 1:1 130 BRANCHIN;
995 1 1:1 132 DEFINEASPECTS;
996 1 1:1 135 NAMEFILES;
997 1 1:1 138 WRITELN('Please be patient');
998 1 1:1 175 WRITELN(' I am starting to sort your datafiles');
999 1 1:1 233 READATTRFILE;
1000 1 1:1 236 SORTATTRFILE;
1001 1 1:1 239 WRITELN(' I Just finished sorting the attributes');
1002 1 1:1 299 READMEASFILE;
1003 1 1:1 302 SORTMEASFILE;
1004 1 1:1 305 WRITELN(' I Just finished sorting the measures');
1005 1 1:1 363 READCOREFILE;
1006 1 1:1 366 SORTCOREFILE;
1007 1 1:1 369 WRITELN(' I Just finished sorting your datafiles');
1008 1 1:1 429 OPENDATAFILE;
1009 1 1:1 432 OPENISSUEINDEX;
1010 1 1:1 435 OPENFASTISSUE;
1011 1 1:1 438 GETUSERSTUFF;
1012 1 1:1 440 TITLEPAGE;
1013 1 1:1 442 REPEAT
1014 1 1:2 442     DISPLAYISSUES;
1015 1 1:2 445     PROCESSISSUE;
1016 1 1:2 447     REPEAT
1017 1 1:3 447         ELIMINATE;
1018 1 1:3 449         PRNTDATASET;
1019 1 1:3 451         PREPKEY(359,'Would you like to remove more performance items from your printout?');
1020 1 1:2 526         UNTIL (ANS='N') OR (ORD(ANS)=27);
1021 1 1:2 539         PREPKEY(360,'Would you like to process another measurement purpose?');
1022 1 1:1 599         UNTIL (ANS='N') OR (ORD(ANS)=27);
1023 1 1:1 614     REWRITE(PRNT,'PRINTER');
1024 1 1:1 635     FOR I:=1 TO 10 DO
1025 1 1:2 652         WRITELN(PRNT,'END OF PRINTOUT FOR ',USERNAME);
1026 1 1:1 714     PAGE(PRNT);
1027 1 1:1 724     CLOSE(PRNT);
1028 1 1:1 733     BRANCHOUT;
1029 1 1:1 735     SETCHAIN('GREETING');
1030 1 1:0 749     END.

```

BEGIN is the main program: 1) sorts attributes and measures, 2) processes measurement purposes, 3) eliminates unwanted performance items, and 4) prints wanted performance items.

```

962 1 20:5 794          IF KEYPRESS THEN
963 1 20:6 801          BEGIN
964 1 20:7 801            READ(ANS);
965 1 20:7 812            IF ORD(ANS)=27 THEN
966 1 20:8 819              BEGIN
967 1 20:9 819                WRITELN(PRNT,'Job cancelled');
968 1 20:9 852                PAGE(PRNT);
969 1 20:9 862                CLOSE(PRNT);
970 1 20:9 871                CLOSE(DATANODE);
971 1 20:9 880                CLOSE(ATTRIBUTES);
972 1 20:9 889                CLOSE(MEASURES);
973 1 20:9 898                EXIT(PRNTDATASET);
974 1 20:8 902                END;
975 1 20:4 902              END;
976 1 20:4 902            END;
977 1 20:2 912            PAGE(PRNT);
978 1 20:2 922            CLOSE(PRNT);
979 1 20:2 931            PREPKEY(303,'Would you like to print another copy of these measurements?');
980 1 20:1 998            UNTIL (ANS='N') OR (ORD(ANS)=27);
981 1 20:1 1011          CLOSE(DATANODE);
982 1 20:1 1020          CLOSE(ATTRIBUTES);
983 1 20:1 1029          CLOSE(MEASURES);
984 1 20:0 1038          END;
985 1 20:0 1068
986 1 20:0 1068
987 1 20:0 1068 (861 05:PRINT2.TEXT8)
988 1 20:0 1068

```

See previous page for program description.

```

989 1 1:0 0 ($SP$)BEGIN
990 1 1:0 0 ($SN-$)
991 1 1:1 0 NISSUES:=5;
992 1 1:1 118 NMEASURES:=400;
993 1 1:1 124 NATTRIBUTES:=200;
994 1 1:1 130 BRANCHIN;
995 1 1:1 132 DEFINEASPECTS;
996 1 1:1 135 NAMEFILES;
997 1 1:1 138 WRITELN('Please be patient');
998 1 1:1 175 WRITELN(' I am starting to sort your datafiles');
999 1 1:1 233 READATTRFILE;
1000 1 1:1 236 SORTATTRFILE;
1001 1 1:1 239 WRITELN(' I just finished sorting the attributes');
1002 1 1:1 299 READMEASFILE;
1003 1 1:1 302 SORTMEASFILE;
1004 1 1:1 305 WRITELN(' I just finished sorting the measures');
1005 1 1:1 363 READCOREFILE;
1006 1 1:1 366 SORTCOREFILE;
1007 1 1:1 369 WRITELN(' I just finished sorting your datafiles');
1008 1 1:1 429 OPENDATAFILE;
1009 1 1:1 432 OPENISSUEINDEX;
1010 1 1:1 435 OPENFASTISSUE;
1011 1 1:1 438 GETUSERSTUFF;
1012 1 1:1 440 TITLEPAGE;
1013 1 1:1 442 REPEAT
1014 1 1:2 442     DISPLAYISSUES;
1015 1 1:2 445     PROCESSISSUE;
1016 1 1:2 447     REPEAT
1017 1 1:3 447         ELIMINATE;
1018 1 1:3 449         PRNTDATASET;
1019 1 1:3 451         PREPKEY(359,'Would you like to remove more performance items from your printout?');
1020 1 1:2 526         UNTIL (ANS='N') OR (ORD(ANS)=27);
1021 1 1:2 539         PREPKEY(360,'Would you like to process another measurement purpose?')
1022 1 1:1 599         UNTIL (ANS='N') OR (ORD(ANS)=27);
1023 1 1:1 614     REWRITE(PRNT,'PRINTER:');
1024 1 1:1 635     FOR I:=1 TO 10 DO
1025 1 1:2 652         WRITELN(PRNT,'END OF PRINTOUT FOR ',USERNAME);
1026 1 1:1 714     PAGE(PRNT);
1027 1 1:1 724     CLOSE(PRNT);
1028 1 1:1 733     BRANCHOUT;
1029 1 1:1 735     SETCHAIN('GREETING');
1030 1 1:0 749     END.

```

BEGIN is the main program: 1) sorts attributes and measures, 2) processes measurement purposes, 3) eliminates unwanted performance items, and 4) prints wanted performance items.

## **PACK**

Pack causes the performance item, attribute and measures data sets to be sorted into numerical order (according to statement number). It also moves unused space to the end of each data set where it becomes available for use with subsequent execution of the PERFITEM and MEASATTR programs.

```

1 1 1:D 1 ($L PRINTER: $)
2 1 1:D 1 ($S+$)
3 1 1:D 1 ($ Program to pack performance items, attribute, and measures lists $)
4 1 1:D 1 ($ Ronald G. Shapiro Version 2.0 10/25/82$)
5 1 1:D 1 Program Packdatasetts;
6 1 1:D 3
7 28 1:D 3
8 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
9 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
10 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
11 28 5:D 1 PROCEDURE SWAPON;
12 28 6:D 1 PROCEDURE SWAPOFF;
13 28 6:D 1
14 1 1:D 1 Uses Chainstuff;
15 1 1:D 3

```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```

16 1 1:D 3 ($P$)TYPE
17 1 1:D 3 ISSUEFILE =RECORD
18 1 1:D 3   NUM:INTEGER;
19 1 1:D 3   NAME:ARRAY[1..2]OF STRING(80);
20 1 1:D 3   DATA:ARRAY[1..225]OF INTEGER(8);
21 1 1:D 3   END;
22 1 1:D 3
23 1 1:D 3 PASSFILE =RECORD
24 1 1:D 3   CURSYS,CURSP,CURSUB,PAC:STRING(80);
25 1 1:D 3   NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
26 1 1:D 3   END;
27 1 1:D 3
28 1 1:D 3 DATABASE =RECORD
29 1 1:D 3   NTAXA: ARRAY[1..4] OF INTEGER;
30 1 1:D 3   TAXA: STRING(80);
31 1 1:D 3   END;
32 1 1:D 3
33 1 1:D 3 FILEATTRIBUTES =RECORD
34 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
35 1 1:D 3   DESCRIPTOR: STRING(68);
36 1 1:D 3   END;
37 1 1:D 3
38 1 1:D 3 FILEMEASURES =RECORD
39 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
40 1 1:D 3   DESCRIPTOR: STRING(68);
41 1 1:D 3   END;
42 1 1:D 3
43 1 1:D 3 TEMPH =RECORD
44 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
45 1 1:D 3   DESCRIPTOR: STRING(68);
46 1 1:D 3   END;
47 1 1:D 3
48 1 1:D 3 TEMPA =RECORD
49 1 1:D 3   NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
50 1 1:D 3   DESCRIPTOR: STRING(68);
51 1 1:D 3   END;
52 1 1:D 3
53 1 1:D 3 TEMPD =RECORD
54 1 1:D 3   NTAXA: ARRAY[1..4] OF INTEGER;
55 1 1:D 3   TAXA: STRING(80);
56 1 1:D 3   END;
57 1 1:D 3

```

ISSUEFILE contains the measurement purposes. PASSFILE passes information between the various programs. DATABASE contains the performance items. FILEATTRIBUTES contains the attributes. FILEMEASURES contains the measures. TEMPM, TEMPA, TEMPD are temporary files used during the pack procedure.



```

58 1 1:D 3 (86P8)VAR
59 1 1:D 3 PASSNODE:FILE OF PASSFILE;
60 1 1:D 474 DATANODE:FILE OF DATABASE;
61 1 1:D 819 COREFILE:FILE OF INTEGER(8);
62 1 1:D 1122 ATTRIBUTES:FILE OF FILEATTRIBUTES;
63 1 1:D 1463 ATTRFILE:FILE OF INTEGER(12);
64 1 1:D 1767 MEASURES:FILE OF FILEMEASURES;
65 1 1:D 2108 MEASFILE:FILE OF INTEGER(12);
66 1 1:D 2412 ISSUE:FILE OF ISSUEFILE;
67 1 1:D 3470 TEMPMEASURES:FILE OF TEMPM;
68 1 1:D 3811 TEMPATTRIBUTES:FILE OF TEMPA;
69 1 1:D 4152 TEMPDATA:FILE OF TEMPD;
70 1 1:D 4497
71 1 1:D 4497 CORE:ARRAY(1..300) OF INTEGER(8);
72 1 1:D 5397 ATTRCORE:ARRAY(1..200) OF INTEGER(12);
73 1 1:D 6197 MEASCORE:ARRAY(1..400) OF INTEGER(12);
74 1 1:D 7797 SCRATCH:ARRAY (1..20) OF INTEGER;
75 1 1:D 7817 ASPECT:ARRAY(1..5) OF STRING(20);
76 1 1:D 7872 HASPECT:ARRAY(1..5) OF INTEGER;
77 1 1:D 7877 CORE2:ARRAY(1..300) OF INTEGER;
78 1 1:D 8177 ATTR2:ARRAY(1..200) OF INTEGER;
79 1 1:D 8377 MEAS2:ARRAY(1..400) OF INTEGER;
80 1 1:D 8777 PRINTIT:ARRAY(1..300) OF BOOLEAN;
81 1 1:D 9077
82 1 1:D 9077 XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING(80);
83 1 1:D 9364 NCURMEASURE,NCURATTRIBUTE,NCURISSUE,NCHARAC,
84 1 1:D 9364 NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
85 1 1:D 9374
86 1 1:D 9374 ISSUENAME,NAMEATCORE,NAMEATTRIBUTES,NAMEHECORE,NAMEHEASURES: STRING(40);
87 1 1:D 9479 NAMETEMPORARY,CORENAME,DATANAME: STRING(40);
88 1 1:D 9542 APHDSK:STRING(10);
89 1 1:D 9548 LEVEL: STRING(10);
90 1 1:D 9554 USERNAME,USERDATE,USERMSG: STRING(80);
91 1 1:D 9677
92 1 1:D 9677 TEMP,CORELAST,T1,T2,T3,T4,T5: INTEGER(8);
93 1 1:D 9698 TEMPX,ATTRLAST,MEASLAST:INTEGER(12);
94 1 1:D 9710
95 1 1:D 9710 NODE,INVERSE,HELP,NSCREEN:INTEGER;
96 1 1:D 9714 NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
97 1 1:D 9717 NISSUES,NUISSUES,NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;

```

These strings, arrays and variables are used by this program.

```

98 1 1:D 9722
99 1 1:D 9722 I,J,K,L,M,N,CUT,INDENT,COUNT,TEMP2:INTEGER;
100 1 1:D 9732
101 1 1:D 9732 REFERENCED, LONGWAY, DONE, OVER, OK, SKIP, NONE:BOOLEAN;
102 1 1:D 9739
103 1 1:D 9739 LINER:STRING(80);
104 1 1:D 9780 LINE:STRING(60);
105 1 1:D 9811
106 1 1:D 9811 ANSWER, REGLINE:STRING(80);
107 1 1:D 9893
108 1 1:D 9893 ANS, ANSHOLD: CHAR;
109 1 1:D 9895
110 1 1:D 9895 PRNT:TEXT;
111 1 1:D 10196
111 1 1:D 10196 (*I 05:UTILITY.TEXT*)
112 1 1:D 10196

```

See previous page for program description.

```

113 1 2:D 1 (%P8)PROCEDURE ANYKEY;
114 1 2:0 0 BEGIN
115 1 2:1 0 WRITELN(' ');
116 1 2:1 18 WRITELN('*** Please press any key to continue ***');
117 1 2:1 78 (%R-8)
118 1 2:1 78 READ(ANS);
119 1 2:1 89 (%R+8)
120 1 2:0 89 END;
121 1 2:0 102

```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```

122 1 3:D 1 (%Ps)PROCEDURE HELPER;
123 1 3:0 0 BEGIN
124 1 3:1 0 WRITELN('For help please refer to your APM MANUAL.');
```

```

125 1 3:0 61 END;
126 1 3:0 74
```

HELPER due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```

127 1 4:D 1 (%P%)PROCEDURE KEYN;
128 1 4:D 1 VAR
129 1 4:D 1 ANSWER: STRING[40];
130 1 4:D 22 II: ARRAY[1..4] OF INTEGER;
131 1 4:D 26 OK: BOOLEAN;
132 1 4:D 27 IIO: INTEGER;
133 1 4:D 28
134 1 4:0 0 BEGIN
135 1 4:0 0 (%R-%)
136 1 4:1 0 REPEAT
137 1 4:2 0 REPEAT
138 1 4:3 0 ANSWER:=
139 1 4:3 27 OK:=TRUE;
140 1 4:3 30 READLN(ANSWER);
141 1 4:3 49 IF LENGTH(ANSWER)=0 THEN
142 1 4:4 57 WRITELN('Please enter the integer again');
143 1 4:2 107 UNTIL LENGTH(ANSWER)<>0;
144 1 4:2 115 IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
145 1 4:3 130 HELPER;
146 1 4:2 132 FOR I:=1 TO 4 DO
147 1 4:3 147 BEGIN
148 1 4:4 147 IIC[I]:=ORD(ANSWER[I])-48;
149 1 4:4 165 IF (IIC[I]<0) OR (IIC[I]>9) THEN
150 1 4:5 192 BEGIN
151 1 4:6 192 IF (I=1) OR (IIC[I]>(ORD(' ')-48)) THEN
152 1 4:7 214 BEGIN
153 1 4:8 214 OK:=FALSE;
154 1 4:8 217 WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
155 1 4:7 275 END;
156 1 4:5 275 END;
157 1 4:3 275 END;
158 1 4:1 285 UNTIL OK=TRUE;
159 1 4:1 292 IIO:=IIC[1];
160 1 4:1 302 FOR I:=2 TO 4 DO
161 1 4:2 317 BEGIN
162 1 4:3 317 IF (IIC[I]>0) AND (IIC[I]<=9) THEN
163 1 4:4 344 IIO:=IIO*10+IIC[I];
164 1 4:2 361 END;
165 1 4:2 371 (%R+%);
166 1 4:1 371 I:=IIO;
167 1 4:0 376 END;
168 1 4:0 398

```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```

169 1 5:D 1 ($P$)PROCEDURE KEY;
170 1 5:D 1 VAR
171 1 5:D 1 I12:INTEGER;
172 1 5:0 0 BEGIN
173 1 5:0 0 ($R-$)
174 1 5:1 0 ANSWER:=
175 1 5:1 27 REPEAT
176 1 5:2 27 READLN(ANSWER);
177 1 5:2 47 ANS:=ANSWER[1];
178 1 5:2 55 IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
179 1 5:2 78 (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<27) THEN
180 1 5:3 98 WRITELN('PLEASE RESPOND YES OR NO!');
181 1 5:2 143 IF (ORD(ANS)>90) THEN
182 1 5:3 150 BEGIN
183 1 5:4 150 I12:=ORD(ANS)-32;
184 1 5:4 157 ANS:=CHR(I12);
185 1 5:3 161 END;
186 1 5:1 161 UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
187 1 5:1 186 ($R+$)
188 1 5:1 186 IF ANS='H' THEN
189 1 5:2 193 HELPER;
190 1 5:0 195 END;
191 1 5:0 210

```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```

192 1 6:0 1 (80P)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
193 1 6:0 0 BEGIN
194 1 6:1 0 HELP:=HLP;
195 1 6:1 9 REPEAT
196 1 6:2 9 WRITE(MSG);
197 1 6:2 20 KEY;
198 1 6:1 22 UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
199 1 6:0 41 END;
200 1 6:0 56

```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplay the message and, once again, waits for a response.

```

201 1 7:0 1 (80P$)PROCEDURE INLINE;
202 1 7:0 1 VAR
203 1 7:0 1 LONGLINE:STRING(125);
204 1 7:0 64 LINEOK:BOOLEAN;
205 1 7:0 65
206 1 7:0 0 BEGIN
207 1 7:1 0 REPEAT
208 1 7:2 0 READLN(LONGLINE);
209 1 7:2 19 LINEOK:=TRUE;
210 1 7:2 22 M:=LENGTH(LONGLINE);
211 1 7:2 29 IF M>80 THEN
212 1 7:3 36 BEGIN
213 1 7:4 36 WRITELN('WARNING LINE CONTAINS OVER 80 CHARACTERS');
214 1 7:4 100 WRITELN(' ');
215 1 7:4 118 PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 80 CHARACTERS? ');
216 1 7:4 166 IF ANS='N' THEN
217 1 7:5 173 BEGIN
218 1 7:6 173 LINEOK:=FALSE;
219 1 7:6 176 WRITELN('PLEASE TYPE LINE AGAIN: ');
220 1 7:5 220 END
221 1 7:4 220 ELSE
222 1 7:5 222 M:=80;
223 1 7:3 226 END;
224 1 7:1 226 UNTIL LINEOK;
225 1 7:1 230 LINER:=COPY(LONGLINE,1,M);
226 1 7:0 248 END;
227 1 7:0 264

```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.



```

228 1 0:0 1 (00P8)PROCEDURE BRANCHIN;
229 1 0:0 0 BEGIN
230 1 0:0 0 (001-8)
231 1 0:1 0 RESET(PASSNODE,'PASSTHRU');
232 1 0:1 18 I:=IORESULT;
233 1 0:1 23 (00148)
234 1 0:1 23 IF I<>0 THEN
235 1 0:2 30 BEGIN
236 1 0:3 30 WRITELN('PASSTHRU FILE DOES NOT EXIST');
237 1 0:3 78 WRITELN(' *****FATAL ERROR*****');
238 1 0:3 123 WRITELN(' ',I);
239 1 0:3 167 ANYKEY;
240 1 0:3 169 SETCHAIN('PGM1');
241 1 0:3 179 EXIT(PROGRAM);
242 1 0:2 183 END;
243 1 0:1 183 GET(PASSNODE);
244 1 0:1 190 CURSYS:=PASSNODE^.CURSYS;
245 1 0:1 198 CURSP:=PASSNODE^.CURSP;
246 1 0:1 206 CURSUB:=PASSNODE^.CURSUB;
247 1 0:1 214 PAC:=PASSNODE^.PAC;
248 1 0:1 220 NCURSYS:=PASSNODE^.NCURSYS;
249 1 0:1 227 NCURSP:=PASSNODE^.NCURSP;
250 1 0:1 234 NCURSUB:=PASSNODE^.NCURSUB;
251 1 0:1 241 NPAC:=PASSNODE^.NPAC;
252 1 0:1 248 CLOSE(PASSNODE);
253 1 0:0 256 END;
254 1 0:0 270

```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```

255 1 9:D 1 (86P)PROCEDURE BRANCHOUT;
256 1 9:0 0 BEGIN
257 1 9:1 0 REWRITE(PASSNODE,'PASSTHRU');
258 1 9:1 20 PASSNODE^.FLAG1:=1;
259 1 9:1 26 PUT(PASSNODE);
260 1 9:1 33 CLOSE(PASSNODE,LOCK);
261 1 9:0 41 END;
262 1 9:0 54
263 1 9:0 54
264 1 9:0 54
265 1 9:0 54 (86I 85:UTILITY.TEXT8)
266 1 9:0 54

```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```

267 1 10:D 1 (%%P)PROCEDURE OPENATTRIBUTESFILE;
268 1 10:0 0 BEGIN
269 1 10:1 0 NATRLAST:=0;
270 1 10:1 4 (%%I-8)
271 1 10:1 4 RESET(ATTRIBUTES,NAMEATTRIBUTES);
272 1 10:1 15 (%%I+8)
273 1 10:1 15 I:=IORESULT;
274 1 10:1 20 IF I<>0 THEN
275 1 10:2 27 BEGIN
276 1 10:3 27 NATRLAST!=-1;
277 1 10:3 32 WRITELN('There is no attributes file on disk');
278 1 10:2 87 END;
279 1 10:1 87 CLOSE(ATTRIBUTES);
280 1 10:0 96 END;
281 1 10:0 108

```

OPENATTRIBUTESFILE checks to see if there is an attributes file on the disk.

```

282 1 11:D 1 ($P*)PROCEDURE OPENMEASURESFILE;
283 1 11:0 0 BEGIN
284 1 11:1 0 NMEASLAST:=0;
285 1 11:1 4 ($I-*)
286 1 11:1 4 RESET(MEASURES,NAMEMEASURES);
287 1 11:1 15 ($I+*)
288 1 11:1 15 I:=IORESULT;
289 1 11:1 20 IF I<>0 THEN
290 1 11:2 27 BEGIN
291 1 11:3 27 WRITELN('There is no measures file on disk');
292 1 11:3 80 NMEASLAST:=-1;
293 1 11:2 85 END;
294 1 11:1 85 CLOSE(MEASURES);
295 1 11:0 94 END;
296 1 11:0 106

```

OPENMEASURESFILE checks to see if there is a measures file on disk.

```

297 1 12:0 1 (80P)PROCEDURE READATTRFILE;
298 1 12:0 0 BEGIN
299 1 12:0 0 (80I-8)
300 1 12:1 0 RESET(ATTRFILE,NAMEATCORE);
301 1 12:1 11 I:=IORESULT;
302 1 12:0 16 (80I+8);
303 1 12:1 16 IF I<>0 THEN
304 1 12:2 23 BEGIN
305 1 12:3 23 REWRITE(ATTRFILE,NAMEATCORE);
306 1 12:3 36 FOR I:=1 TO MATTRIBUTES DO
307 1 12:4 52 BEGIN
308 1 12:5 52 ATTRCORE[I]:=0;
309 1 12:5 79 ATTRFILE^:=ATTRCORE[I];
310 1 12:5 107 PUT(ATTRFILE);
311 1 12:5 115 IF EOF(ATTRFILE) THEN
312 1 12:6 125 BEGIN
313 1 12:7 125 WRITELN('OUT OF DISK SPACE');
314 1 12:7 162 ANYKEY;
315 1 12:6 164 END;
316 1 12:4 164 END;
317 1 12:3 174 ATTRLAST:=0;
318 1 12:3 189 MATTRLAST:=0;
319 1 12:3 193 ATTRFILE^:=ATTRLAST;
320 1 12:3 209 PUT(ATTRFILE);
321 1 12:3 217 CLOSE(ATTRFILE,LOCK);
322 1 12:2 226 END
323 1 12:1 226 ELSE
324 1 12:2 228 BEGIN
325 1 12:3 228 FOR I:=1 TO MATTRIBUTES DO
326 1 12:4 244 BEGIN
327 1 12:5 244 GET(ATTRFILE);
328 1 12:5 252 ATTRCORE[I]:=ATTRFILE^;
329 1 12:4 280 END;
330 1 12:3 290 GET(ATTRFILE);
331 1 12:3 298 ATTRLAST:=ATTRFILE^;
332 1 12:3 314 MATTRLAST:=TRUNC(ATTRLAST);
333 1 12:3 327 CLOSE(ATTRFILE);
334 1 12:2 336 END;
335 1 12:0 336 END;
336 1 12:0 354

```

READATTRFILE reads the index to the attributes file from the disk file ATTRFILE and places it into the array ATTRCORE.

```

337 1 13:0 1 (80P8)PROCEDURE READMEASFILE;
338 1 13:0 0 BEGIN
339 1 13:0 0 (801-8)
340 1 13:1 0 RESET(MEASFILE,NAMECORE);
341 1 13:1 11 I:=IORESULT;
342 1 13:0 16 (801+8);
343 1 13:1 16 IF I<>0 THEN
344 1 13:2 23 BEGIN
345 1 13:3 23 REWRITE(MEASFILE,NAMECORE);
346 1 13:3 36 FOR I:=1 TO NMEASURES DO
347 1 13:4 52 BEGIN
348 1 13:5 52 MEASCORE[I]:=0;
349 1 13:5 79 MEASFILE^:=MEASCORE[I];
350 1 13:5 107 PUT(MEASFILE);
351 1 13:5 115 IF EOF(MEASFILE) THEN
352 1 13:6 125 BEGIN
353 1 13:7 125 WRITELN('OUT OF DISK SPACE');
354 1 13:7 162 ANYKEY;
355 1 13:6 164 END;
356 1 13:4 164 END;
357 1 13:3 174 MEASLAST:=0;
358 1 13:3 189 NMEASLAST:=0;
359 1 13:3 193 MEASFILE^:=MEASLAST;
360 1 13:3 209 PUT(MEASFILE);
361 1 13:3 217 CLOSE(MEASFILE,LOCK);
362 1 13:2 226 END
363 1 13:1 226 ELSE
364 1 13:2 228 BEGIN
365 1 13:3 228 FOR I:=1 TO NMEASURES DO
366 1 13:4 244 BEGIN
367 1 13:5 244 GET(MEASFILE);
368 1 13:5 252 MEASCORE[I]:=MEASFILE^;
369 1 13:4 280 END;
370 1 13:3 290 GET(MEASFILE);
371 1 13:3 298 MEASLAST:=MEASFILE^;
372 1 13:3 314 NMEASLAST:=TRUNC(MEASLAST);
373 1 13:3 327 CLOSE(MEASFILE);
374 1 13:2 336 END;
375 1 13:0 336 END;
376 1 13:0 354

```

READMEASFILE reads the index to the measures file from the disk file MEASFILE and places it into the array MEASCORE.

```

377 1 14:0 1 (84P)PROCEDURE CLOSECOREFILE;
378 1 14:0 0 BEGIN
379 1 14:1 0 RESET(COREFILE,CORENAME);
380 1 14:1 13 FOR I:=1 TO 300 DO
381 1 14:2 29 BEGIN
382 1 14:3 29 COREFILE^:=CORE(I);
383 1 14:3 57 PUT(COREFILE);
384 1 14:2 45 END;
385 1 14:1 75 CORELAST:=NCORELAST;
386 1 14:1 92 COREFILE^:=CORELAST;
387 1 14:1 108 PUT(COREFILE);
388 1 14:1 116 CLOSE(COREFILE);
389 1 14:0 125 END;
390 1 14:0 140

```

CLOSECOREFILE copies the index to the performance items from the array core to the disk file COREFILE.

```

391 1 15:D 1 (86P)PROCEDURE CLOSEATTRFILE;
392 1 15:0 0 BEGIN
393 1 15:1 0 RESET(ATTRFILE,NAMEATCORE);
394 1 15:1 13 FOR I:=1 TO MATTRIBUTES DO
395 1 15:2 29 BEGIN
396 1 15:3 29 ATTRFILE^:=ATTRCORE[I];
397 1 15:3 57 PUT(ATTRFILE);
398 1 15:2 45 END;
399 1 15:1 75 ATTRLAST:=MATTRLAST;
400 1 15:1 92 ATTRFILE^:=ATTRLAST;
401 1 15:1 108 PUT(ATTRFILE);
402 1 15:1 116 CLOSE(ATTRFILE);
403 1 15:0 125 END;
404 1 15:0 140

```

CLOSEATTRFILE copies the index to the attribute file from the array ATTRCORE to the disk file ATTRFILE.



```

405 1 16:B 1 (88P)PROCEDURE CLOSEMEASFILE;
406 1 16:0 0 BEGIN
407 1 16:1 0 RESET(MEASFILE,NAMECORE);
408 1 16:1 13 FOR I:=1 TO NMEASURES DO
409 1 16:2 29 BEGIN
410 1 16:3 29 MEASFILE^:=MEASCORE[I];
411 1 16:3 57 PUT(MEASFILE);
412 1 16:2 65 END;
413 1 16:1 75 MEASLAST:=NMEASLAST;
414 1 16:1 92 MEASFILE^:=MEASLAST;
415 1 16:1 108 PUT(MEASFILE);
416 1 16:1 116 CLOSE(MEASFILE);
417 1 16:0 125 END;
418 1 16:0 140

```

CLOSEMEASFILE copies the index to the measures file from the array MEASCORE to the disk file MEASFILE.

```

419 1 17:0 1 (86P) PROCEDURE OPENDATAFILE;
420 1 17:0 0 BEGIN
421 1 17:1 0 NCORELAST:=0;
422 1 17:1 4 (86I-8)
423 1 17:1 4 RESET(DATANODE,DATANAME);
424 1 17:1 15 (86I+8)
425 1 17:1 15 I:=IORESULT;
426 1 17:1 20 IF I<>0 THEN
427 1 17:2 27 BEGIN
428 1 17:3 27 WRITELN('There is no performance items file on disk');
429 1 17:3 89 NCORELAST:=-1;
430 1 17:2 94 END;
431 1 17:1 94 CLOSE(DATANODE);
432 1 17:0 103 END;
433 1 17:0 116

```

OPENDATAFILE determines whether there are any performance items on the disk.

```

434 1 18:0 1 (80P)PROCEDURE DEFINEASPECTS;
435 1 18:0 0 BEGIN
436 1 18:1 0 ASPECT(1):='Potentialities';
437 1 18:1 30 ASPECT(2):='Processes';
438 1 18:1 55 ASPECT(3):='Products';
439 1 18:1 79 ASPECT(4):='Environment';
440 1 18:1 104 ASPECT(5):='Constraints';
441 1 18:1 133 NASPECT(1):=1;
442 1 18:1 146 NASPECT(2):=2;
443 1 18:1 159 NASPECT(3):=3;
444 1 18:1 172 NASPECT(4):=4;
445 1 18:1 185 NASPECT(5):=5;
446 1 18:0 198 END;
447 1 18:0 210

```

DEFINEASPECTS tells computer the name assigned to each of the aspects.

```

448 1 19:0 1 (80P)PROCEDURE READCOREFILE;
449 1 19:0 0 BEGIN
450 1 19:0 0 (80I-8)
451 1 19:1 0 RESET(COREFILE,CORENAME);
452 1 19:1 11 I:=IORESULT;
453 1 19:1 16 (80I+8)
454 1 19:1 16 IF I<>0 THEN
455 1 19:2 23 BEGIN
456 1 19:3 23 IF I=9 THEN
457 1 19:4 30 BEGIN
458 1 19:5 30 PAGE(OUTPUT);
459 1 19:5 40 WRITELN('THE APMDISK IS NOT MOUNTED');
460 1 19:5 86 WRITELN('');
461 1 19:5 106 WRITELN('PLEASE PLACE IT IN DRIVE #2');
462 1 19:5 153 ANYKEY;
463 1 19:5 155 READCOREFILE;
464 1 19:5 157 EXIT(READCOREFILE)
465 1 19:4 161 END
466 1 19:3 161 ELSE
467 1 19:4 163 BEGIN
468 1 19:5 163 WRITELN('COREFILE DOES NOT EXIST');
469 1 19:5 206 WRITELN(' ****FATAL ERROR**** ');
470 1 19:5 249 WRITELN(' ',I);
471 1 19:5 292 ANYKEY;
472 1 19:5 294 BRANCHOUT;
473 1 19:5 296 SETCHAIN('GREETING');
474 1 19:5 310 EXIT(PROGRAM);
475 1 19:4 314 END;
476 1 19:2 314 END
477 1 19:1 314 ELSE
478 1 19:2 316 FOR I:=1 TO 300 DO
479 1 19:3 332 BEGIN
480 1 19:4 332 GET(COREFILE);
481 1 19:4 340 CORE(I):=COREFILE^;
482 1 19:3 368 END;
483 1 19:1 378 GET(COREFILE);
484 1 19:1 386 CORELAST:=COREFILE^;
485 1 19:1 402 NCORELAST:=TRUNC(CORELAST);
486 1 19:1 415 CLOSE(COREFILE)
487 1 19:0 424 END;
488 1 19:0 444

```

READCOREFILE copies performance items from the disk file DATANODE to the CORE file.

```

489 1 20:D 1 (88P)PROCEDURE SORTATTRFILE;
490 1 20:0 0 BEGIN
491 1 20:1 0 IF MATTRLAST<2 THEN
492 1 20:2 7 EXIT(SORTATTRFILE);
493 1 20:1 11 FOR I:=1 TO MATTRIBUTES DO
494 1 20:2 27 ATTR2[I]:=I;
495 1 20:1 54 I:=2;
496 1 20:1 60 REPEAT
497 1 20:2 60 IF ATTRCORE[I]<ATTRCORE[I-1] THEN
498 1 20:3 105 BEGIN
499 1 20:4 105 TEMPX:=ATTRCORE[I];
500 1 20:4 133 ATTRCORE[I]:=ATTRCORE[I-1];
501 1 20:4 175 ATTRCORE[I-1]:=TEMPX;
502 1 20:4 205 TEMP2:=ATTR2[I];
503 1 20:4 224 ATTR2[I]:=ATTR2[I-1];
504 1 20:4 258 ATTR2[I-1]:=TEMP2;
505 1 20:4 279 IF I>2 THEN
506 1 20:5 284 I:=I-1;
507 1 20:3 294 END
508 1 20:2 294 ELSE
509 1 20:3 296 I:=I+1;
510 1 20:1 304 UNTIL I>MATTRLAST;
511 1 20:0 313 END;
512 1 20:0 332

```

SORTATTRFILE forms an array ATTR2 which contains a sorted permutation vecyor referencing the attributes file [sorted by numerical value of the index]--sort attributes index into assending numerical order.

```

513 1 21:0 1 (80P)PROCEDURE SORTMEASFILE;
514 1 21:0 0 BEGIN
515 1 21:1 0 IF NMEASLAST<2 THEN
516 1 21:2 7 EXIT(SORTMEASFILE);
517 1 21:1 11 FOR I:=1 TO NMEASURES DO
518 1 21:2 27 MEAS2[I]:=I;
519 1 21:1 56 I:=2;
520 1 21:1 60 REPEAT
521 1 21:2 60 IF MEASCORE[I]<MEASCORE[I-1] THEN
522 1 21:3 105 BEGIN
523 1 21:4 105 TEMPX:=MEASCORE[I];
524 1 21:4 133 MEASCORE[I]:=MEASCORE[I-1];
525 1 21:4 175 MEASCORE[I-1]:=TEMPX;
526 1 21:4 205 TEMP2:=MEAS2[I];
527 1 21:4 224 MEAS2[I]:=MEAS2[I-1];
528 1 21:4 258 MEAS2[I-1]:=TEMP2;
529 1 21:4 279 IF I>2 THEN
530 1 21:5 286 I:=I-1;
531 1 21:3 294 END
532 1 21:2 294 ELSE
533 1 21:3 294 I:=I+1;
534 1 21:1 304 UNTIL I>NMEASLAST;
535 1 21:0 313 END;
536 1 21:0 332

```

SORTMEASFILE forms an array MEAS2 which contains a sorted permutation vector referencing the measures file—sort measures index into ascending numerical order.

```

537 1 22:0 1 (##P*)PROCEDURE SORTCOREFILE;
538 1 22:0 0 BEGIN
539 1 22:1 0 IF NCORELAST<2 THEN
540 1 22:2 7 EXIT(SORTCOREFILE);
541 1 22:1 11 FOR I:=1 TO 300 DO
542 1 22:2 27 CORE2[I]:=I;
543 1 22:1 56 I:=2;
544 1 22:1 60 REPEAT
545 1 22:2 60 IF CORE[I]<CORE[I-1] THEN
546 1 22:3 105 BEGIN
547 1 22:4 105 TEMP:=CORE[I];
548 1 22:4 133 CORE[I]:=CORE[I-1];
549 1 22:4 175 CORE[I-1]:=TEMP;
550 1 22:4 205 TEMP2:=CORE2[I];
551 1 22:4 224 CORE2[I]:=CORE2[I-1];
552 1 22:4 258 CORE2[I-1]:=TEMP2;
553 1 22:4 279 IF I>2 THEN
554 1 22:5 286 I:=I-1;
555 1 22:3 294 END
556 1 22:2 294 ELSE
557 1 22:3 296 I:=I+1;
558 1 22:1 304 UNTIL I>NCORELAST;
559 1 22:0 313 END;
560 1 22:0 332

```

SORTCOREFILE forms an array CORE2 which contains a sorted permutation vector referencing the core file.

```

561 1 23:D 1 (**)PROCEDURE COPYATTRIBUTES;
562 1 23:0 0 BEGIN
563 1 23:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
564 1 23:1 13 REWRITE(TEMPATTRIBUTES,NAMETEMPORARY);
565 1 23:1 26 IF IORESULT<>0 THEN
566 1 23:2 32 BEGIN
567 1 23:3 32 WRITELN('PROBLEM CREATING TEMPORARY DATASET');
568 1 23:3 86 ANYKEY;
569 1 23:2 88 END;
570 1 23:1 88 FOR I:=1 TO NATTRLAST DO
571 1 23:2 104 BEGIN
572 1 23:3 104 SEEK(ATTRIBUTES,ATTR2[I]);
573 1 23:3 128 GET(ATTRIBUTES);
574 1 23:3 136 SEEK(TEMPATTRIBUTES,I);
575 1 23:3 147 TEMPATTRIBUTES^:=ATTRIBUTES^;
576 1 23:3 155 PUT(TEMPATTRIBUTES);
577 1 23:2 163 END;
578 1 23:1 173 FOR I:=1 TO NATTRLAST DO
579 1 23:2 189 BEGIN
580 1 23:3 189 SEEK(TEMPATTRIBUTES,I);
581 1 23:3 200 GET(TEMPATTRIBUTES);
582 1 23:3 208 SEEK(ATTRIBUTES,I);
583 1 23:3 219 ATTRIBUTES^:=TEMPATTRIBUTES^;
584 1 23:3 227 PUT(ATTRIBUTES);
585 1 23:2 235 END;
586 1 23:1 245 CLOSE(ATTRIBUTES);
587 1 23:1 254 CLOSE(TEMPATTRIBUTES);
588 1 23:0 263 END;
589 1 23:0 280

```

COPYATTRIBUTES copies the attributes file from the disk file to the temporary storage disk [sorting attributes into numerical order] and temporary disk back to the usual storage disk.



```

590 1 24:0 1 (**P*)PROCEDURE COPYMEASURES;
591 1 24:0 0 BEGIN
592 1 24:1 0 RESET(MEASURES,NAMEMEASURES);
593 1 24:1 13 REWRITE(TEMPMEASURES,NAMETEMPORARY);
594 1 24:1 26 IF IORESULT<>0 THEN
595 1 24:2 32 BEGIN
596 1 24:3 32 WRITELN('PROBLEM CREATING TEMPORARY DATASET');
597 1 24:3 86 ANYKEY;
598 1 24:2 88 END;
599 1 24:1 88 FOR I:=1 TO NMEASLAST DO
600 1 24:2 104 BEGIN
601 1 24:3 104 SEEK(MEASURES,MEAS2(I));
602 1 24:3 128 GET(MEASURES);
603 1 24:3 136 SEEK(TEMPMEASURES,I);
604 1 24:3 147 TEMPMEASURES^:=MEASURES^;
605 1 24:3 155 PUT(TEMPMEASURES);
606 1 24:2 163 END;
607 1 24:1 173 FOR I:=1 TO NMEASLAST DO
608 1 24:2 189 BEGIN
609 1 24:3 189 SEEK(TEMPMEASURES,I);
610 1 24:3 200 GET(TEMPMEASURES);
611 1 24:3 208 SEEK(MEASURES,I);
612 1 24:3 219 MEASURES^:=TEMPMEASURES^;
613 1 24:3 227 PUT(MEASURES);
614 1 24:2 235 END;
615 1 24:1 245 CLOSE(MEASURES);
616 1 24:1 254 CLOSE(TEMPMEASURES);
617 1 24:0 263 END;
618 1 24:0 280

```

COPYMEASURES copies the measures file from the disk file to the temporary storage disk [sorting measures into numeric order ] and copying the items from the temporary disk back to the usual storage disk.

```

619 1 25:0 1 (**)PROCEDURE COPYCORE;
620 1 25:0 0 BEGIN
621 1 25:1 0 RESET(DATANODE,DATANAME);
622 1 25:1 13 REWRITE(TEMPDATA,NAMETEMPORARY);
623 1 25:1 26 IF IORESULT<>0 THEN
624 1 25:2 32 BEGIN
625 1 25:3 32 WRITELN('PROBLEM CREATING TEMPORARY DATASET');
626 1 25:3 84 ANYKEY;
627 1 25:2 88 END;
628 1 25:1 88 FOR I:=1 TO NCORELAST DO
629 1 25:2 104 BEGIN
630 1 25:3 104 SEEK(DATANODE,CORE2[I]);
631 1 25:3 128 GET(DATANODE);
632 1 25:3 136 SEEK(TEMPDATA,I);
633 1 25:3 147 TEMPDATA^:=DATANODE^;
634 1 25:3 155 PUT(TEMPDATA);
635 1 25:2 163 END;
636 1 25:1 173 FOR I:=1 TO NCORELAST DO
637 1 25:2 189 BEGIN
638 1 25:3 189 SEEK(TEMPDATA,I);
639 1 25:3 200 GET(TEMPDATA);
640 1 25:3 208 SEEK(DATANODE,I);
641 1 25:3 219 DATANODE^:=TEMPDATA^;
642 1 25:3 227 PUT(DATANODE);
643 1 25:2 235 END;
644 1 25:1 245 CLOSE(DATANODE);
645 1 25:1 254 CLOSE(TEMPDATA);
646 1 25:0 263 END;
647 1 25:0 280

```

COPYCORE copies the core file from the disk file to the temporary storage disk [sorting items into numeric order] and copying the items from the temporary disk back to the usual storage disk.

```

648 1 26:D 1 (**P*)PROCEDURE REMOVEATTRIBUTES;
649 1 26:0 0 BEGIN
650 1 26:1 0 SEEK(ATTRIBUTES,J);
651 1 26:1 11 FOR L:=1 TO 6 DO
652 1 26:2 25 ATTRIBUTES^.NDESCRIPTOR(L):=0;
653 1 26:1 50 ATTRIBUTES^.DESCRIPTOR:='';
654 1 26:1 60 PUT(ATTRIBUTES);
655 1 26:1 68 ATTRCORE(J):=0;
656 1 26:0 95 END;
657 1 26:0 110

```

REMOVEATTRIBUTES removes an attribute from attributes file.

```

658 1 27:0 1 (86P8)PROCEDURE REMOVE MEASURES;
659 1 27:0 0 BEGIN
660 1 27:1 0 SEEK(MEASURES,J);
661 1 27:1 11 FOR L:=1 TO 6 DO
662 1 27:2 25 MEASURES^.NDESCRIPTOR[L]:=0;
663 1 27:1 50 MEASURES^.DESCRIPTOR:='';
664 1 27:1 60 PUT(MEASURES);
665 1 27:1 68 MEASURES[J]:=0;
666 1 27:0 95 END;
667 1 27:0 110

```

REMOVE MEASURES removes a measure from measures file.

```

668 1 28:0 1 (SSP)PROCEDURE REMOVEDATA;
669 1 28:0 0 BEGIN
670 1 28:1 0 SEEK(DATANODE,J);
671 1 28:1 11 FOR L:=1 TO 4 DO
672 1 28:2 25 DATANODE^.NTAXAELJ:=0;
673 1 28:1 50 DATANODE^.TAXA:='';
674 1 28:1 60 PUT(DATANODE);
675 1 28:1 68 COREIJJ:=0;
676 1 28:0 95 END;
677 1 28:0 110

```

REMOVEDATA removes a performance item from data file.

```

678 1 29:0 1 (86P8)PROCEDURE COMPACTATTRIBUTES;
679 1 29:0 0 BEGIN
680 1 29:1 0 RESET(ATTRIBUTES,NAMEATTRIBUTES);
681 1 29:1 13 N:=0;
682 1 29:1 17 I:=0;
683 1 29:1 21 REPEAT
684 1 29:2 21 I:=I+1;
685 1 29:2 29 REPEAT
686 1 29:3 29 IF ATTRCORE[I+N]=0 THEN
687 1 29:4 63 N:=N+1;
688 1 29:3 71 J:=I+N;
689 1 29:3 81 IF J>MATRLAST THEN
690 1 29:4 90 BEGIN
691 1 29:5 90 I:=I+1;
692 1 29:5 98 FOR J:=I TO MATRLAST DO
693 1 29:6 116 REMOVEATTRIBUTES;
694 1 29:5 128 MATRLAST:=MATRLAST-N;
695 1 29:5 138 CLOSE(ATTRIBUTES);
696 1 29:5 147 EXIT(COMPACTATTRIBUTES);
697 1 29:4 151 END;
698 1 29:2 151 UNTIL ATTRCORE[I+N]<>0;
699 1 29:2 185 ATTRCORE[I]:=ATTRCORE[I+N];
700 1 29:2 229 J:=I+N;
701 1 29:2 239 SEEK(ATTRIBUTES,J);
702 1 29:2 250 GET(ATTRIBUTES);
703 1 29:2 258 SEEK(ATTRIBUTES,I);
704 1 29:2 269 PUT(ATTRIBUTES);
705 1 29:1 277 UNTIL J=MATRLAST;
706 1 29:1 286 I:=I+1;
707 1 29:1 294 FOR J:=I TO MATRLAST DO
708 1 29:2 312 REMOVEATTRIBUTES;
709 1 29:1 324 MATRLAST:=MATRLAST-N;
710 1 29:1 334 CLOSE(ATTRIBUTES);
711 1 29:0 343 END;
712 1 29:0 344

```

COMPACTATTRIBUTES packs attribute data set so that all blank entries are pushed to the end of the data set.

```

713 1 30:8 1 (88P*)PROCEDURE COMPACTMEASURES;
714 1 30:0 0 BEGIN
715 1 30:1 0 RESET(MEASURES,NAMEMEASURES);
716 1 30:1 13 M:=0;
717 1 30:1 17 I:=0;
718 1 30:1 21 REPEAT
719 1 30:2 21 I:=I+1;
720 1 30:2 29 REPEAT
721 1 30:3 29 IF MEASCORE[I+M]=0 THEN
722 1 30:4 43 M:=M+1;
723 1 30:3 71 J:=I+M;
724 1 30:3 81 IF J>NMEASLAST THEN
725 1 30:4 90 BEGIN
726 1 30:5 90 I:=I+1;
727 1 30:5 98 FOR J:=I TO NMEASLAST DO
728 1 30:6 116 REMOVEMEASURES;
729 1 30:5 128 NMEASLAST:=NMEASLAST-M;
730 1 30:5 138 CLOSE(MEASURES);
731 1 30:5 147 EXIT(COMPACTMEASURES);
732 1 30:4 151 END;
733 1 30:2 151 UNTIL MEASCORE[I+M]<>0;
734 1 30:2 185 MEASCORE[I]:=MEASCORE[I+M];
735 1 30:2 229 J:=I+M;
736 1 30:2 239 SEEK(MEASURES,J);
737 1 30:2 250 GET(MEASURES);
738 1 30:2 258 SEEK(MEASURES,I);
739 1 30:2 269 PUT(MEASURES);
740 1 30:1 277 UNTIL J=NMEASLAST;
741 1 30:1 286 I:=I+1;
742 1 30:1 294 FOR J:=I TO NMEASLAST DO
743 1 30:2 312 REMOVEMEASURES;
744 1 30:1 324 NMEASLAST:=NMEASLAST-M;
745 1 30:1 334 CLOSE(MEASURES);
746 1 30:0 343 END;
747 1 30:0 344

```

COMPACTMEASURES packs measures data set more efficiently so that all blank entries are pushed to the end of the data set.

```

748 1 31:0 1 ($P8)PROCEDURE COMPACTCORE;
749 1 31:0 0 BEGIN
750 1 31:1 0 RESET(DATANODE,DATANAME);
751 1 31:1 13 M:=0;
752 1 31:1 17 I:=0;
753 1 31:1 21 REPEAT
754 1 31:2 21 I:=I+1;
755 1 31:2 29 REPEAT
756 1 31:3 29 IF CORE[I+M]=0 THEN
757 1 31:4 63 M:=M+1;
758 1 31:3 71 J:=I+M;
759 1 31:3 81 IF J>NCORELAST THEN
760 1 31:4 90 BEGIN
761 1 31:5 90 I:=I+1;
762 1 31:5 98 FOR J:=I TO NCORELAST DO
763 1 31:6 116 REMOVEDATA;
764 1 31:5 128 NCORELAST:=NCORELAST-M;
765 1 31:5 138 CLOSE(DATANODE);
766 1 31:5 147 EXIT(COMPACTCORE);
767 1 31:4 151 END;
768 1 31:2 151 UNTIL CORE[I+M]<>0;
769 1 31:2 185 CORE[J]:=CORE[I+M];
770 1 31:2 229 J:=I+M;
771 1 31:2 239 SEEK(DATANODE,J);
772 1 31:2 250 GET(DATANODE);
773 1 31:2 258 SEEK(DATANODE,I);
774 1 31:2 269 PUT(DATANODE);
775 1 31:1 277 UNTIL J=NCORELAST;
776 1 31:1 286 I:=I+1;
777 1 31:1 294 FOR J:=I TO NCORELAST DO
778 1 31:2 312 REMOVEDATA;
779 1 31:1 324 NCORELAST:=NCORELAST-M;
780 1 31:1 334 CLOSE(DATANODE);
781 1 31:0 343 END;
782 1 31:0 344

```

COMPACTCORE packs performance item data set more efficiently so that all blank entries are pushed to the end of the data set.



```

783 1 32:0 1 (SP)PROCEDURE DONOT;
784 1 32:0 0 BEGIN
785 1 32:1 0 WRITELN(' This file contains less than 2 items, thus it will not be packed');
786 1 32:0 84 END;
787 1 32:0 98

```

DONOT displays warning message that file will not be packed.

```

788 1 33:D 1 (80P*)PROCEDURE PROPERUTLDISK;
789 1 33:0 0 BEGIN
790 1 33:1 0 REPEAT
791 1 33:1 0 (80I-8)
792 1 33:2 0 RESET(TEMPDATA,'APMUTL:TEMPORARY');
793 1 33:2 27 (80I+8)
794 1 33:2 27 K:=IDRESULT;
795 1 33:2 32 IF K=0 THEN
796 1 33:3 39 CLOSE(TEMPDATA);
797 1 33:2 48 IF K=9 THEN
798 1 33:3 55 BEGIN
799 1 33:4 55 WRITELN('Please place the APM UTILITY disk in drive 0 1');
800 1 33:4 121 ANYKEY;
801 1 33:3 123 END;
802 1 33:1 123 UNTIL K<>9;
803 1 33:0 130 END;
804 1 33:0 144

```

PROPERUTLDISK checks to be sure APMUTL (the disk used for temporary storage) is in Drive #1.

```

805 1 34:0 1 (80P)PROCEDURE PROPERMAINDISK;
806 1 34:0 0 BEGIN
807 1 34:1 0 REPEAT
808 1 34:1 0 (80I-8)
809 1 34:2 0 RESET(TEMPDATA,'APMSYS:TEMPORARY');
810 1 34:2 27 (80I+8)
811 1 34:2 27 K:=IORESULT;
812 1 34:2 32 IF K=0 THEN
813 1 34:3 39 CLOSE(TEMPDATA);
814 1 34:2 48 IF K=9 THEN
815 1 34:3 55 BEGIN
816 1 34:4 55 WRITELN('Please place the APM SYSTEM disk in drive # 1');
817 1 34:4 120 ANYKEY;
818 1 34:3 122 END;
819 1 34:1 122 UNTIL K<>9;
820 1 34:0 129 END;
821 1 34:0 144

```

PROPERMAINDISK checks to be sure the APYSYS disk has been returned to Drive #1 before returning to select a different analytic procedure.

```

822 135:D 1 (80P)PROCEDURE ASSIGNNAMES;
823 135:0 0 BEGIN
824 135:1 0 APHDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
825 135:1 84 NAMETEMPORARY:=CONCAT('APHUTL:TEMPORARY');
826 135:1 121 NAMEATCORE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'AC');
827 135:1 217 NAMEATTRIBUTES:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),
'AT');
828 135:1 313 NAMEHCORE:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'HC');
829 135:1 409 NAMEHEASURES:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),
'HE');
830 135:1 505 CORENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
831 135:1 601 DATANAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
832 135:1 697 ISSUENAME:=CONCAT(APHDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
833 135:0 793 END;
834 135:0 806

```

ASSIGNNAMES determines file names based upon system class, system and subsystem names.

```

835 1 1:0 0 (##P#)BEGIN
836 1 1:0 0 (##N#)
837 1 1:1 0 PAGE(OUTPUT);
838 1 1:1 141 BRANCHIN;
839 1 1:1 143 PROPERUTLDISK;
840 1 1:1 145 WRITELN('I am going to sort and peck all data sets, but I am slow',chr(13),
841 1 1:1 224 ' so please take a coffee break at this time',chr(13),chr(13));
842 1 1:1 309 NISSUES:=5;
843 1 1:1 313 NMEASURES:=400;
844 1 1:1 319 NATTRIBUTES:=200;
845 1 1:1 323
846 1 1:1 325 ASSIGNNAMES;
847 1 1:1 327
848 1 1:1 327 WRITELN('Processing attributes');
849 1 1:1 368 OPENATTRIBUTESFILE;
850 1 1:1 370 IF NATTRLAST<>-1 THEN
851 1 1:2 378 BEGIN
852 1 1:3 378 READATTRFILE;
853 1 1:3 380 IF NATTRLAST>2 THEN
854 1 1:4 387 BEGIN
855 1 1:5 387 COMPACTATTRIBUTES;
856 1 1:5 389 SORTATTRFILE;
857 1 1:5 391 COPYATTRIBUTES;
858 1 1:5 393 CLOSEATTRFILE;
859 1 1:4 395 END
860 1 1:3 395 ELSE
861 1 1:4 397 DONOT;
862 1 1:2 399 END;
863 1 1:2 399
864 1 1:1 399 WRITELN('Processing measures');
865 1 1:1 438 OPENMEASURESFILE;
866 1 1:1 440 IF NMEASLAST<>-1 THEN
867 1 1:2 448 BEGIN
868 1 1:3 448 READMEASFILE;
869 1 1:3 450 IF NMEASLAST>2 THEN
870 1 1:4 457 BEGIN
871 1 1:5 457 COMPACTMEASURES;
872 1 1:5 459 SORTMEASFILE;
873 1 1:5 461 COPYMEASURES;
874 1 1:5 463 CLOSEMEASFILE;

```

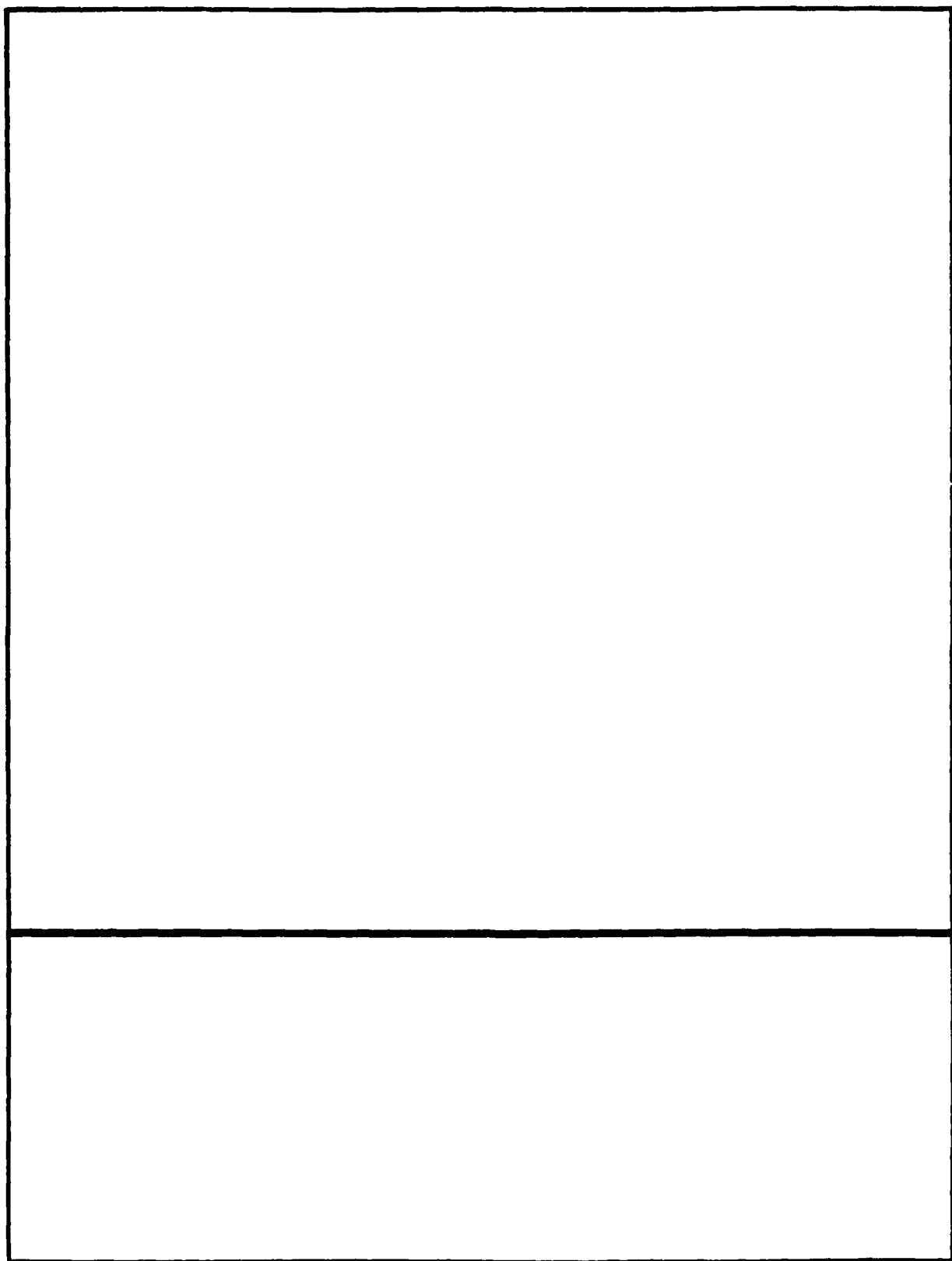
Main Program: Governs overall packing and sorting of attributes, measures and performance items.

```

875 1 114 465      END
876 1 113 465      ELSE
877 1 114 467      DONOT;
878 1 112 469      END;
879 1 112 469
880 1 111 469      Writeln('Processing performance items');
881 1 111 517      OPENDATAFILE;
882 1 111 519      IF NCORELAST<>-1 THEN
883 1 112 527      BEGIN
884 1 113 527      READCOREFILE;
885 1 113 529      IF NCORELAST>2 THEN
886 1 114 536      BEGIN
887 1 115 536      COMPACTCORE;
888 1 115 538      SORTCOREFILE;
889 1 115 540      COPYCORE;
890 1 115 542      CLOSECOREFILE;
891 1 114 544      END
892 1 113 544      ELSE
893 1 114 546      DONOT;
894 1 112 548      END;
895 1 112 548
896 1 111 548      Writeln(chr(13),'Data sets are packed and sorted',chr(13),chr(13),
897 1 111 621      '....so long for now');
898 1 111 660
899 1 111 660      PROPERMAINDISK;
900 1 111 662
901 1 111 662      BRANCHOUT;
902 1 111 664
903 1 111 664      SETCHAIN('GREETING');
904 1 111 678
905 1 110 678      END.

```

See previous page for program description.



## **MISCELLANEOUS**

Two of the programs in this system are present on the BOOT disk. They ask the analyst to set up the APM System disk. The remaining programs in this section are not actually part of the APM Demonstration Package. They are used to prepare data sets for use with the APM System.



```

1 1 1:D 1 ($L PRINTER:*)
2 1 1:D 1 ($SYSTEM.STARTUP TELLS THE ANALYST TO PLACE THE REAL APM SYSTEM DISK IN DRIVE # 1*)
3 1 1:D 1 ($RONALD G. SHAPIRO V 2.0 10/25/82*)
4 1 1:D 1
5 1 1:D 1 PROGRAM STARTUP;
6 28 1:D 3
7 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
8 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
9 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
10 28 5:D 1 PROCEDURE SWAPON;
11 28 6:D 1 PROCEDURE SWAPOFF;
12 28 6:D 1
13 1 1:D 1 USES CHAINSTUFF;
14 1 1:D 3 VAR
15 1 1:D 3 X:CHAR;
16 1 1:D 0 BEGIN
17 1 1:1 0 PAGE(OUTPUT);
18 1 1:1 15 WRITELN('Please insert the APM SYSTEM DISK in drive # 1');
19 1 1:1 81 WRITELN(' Then press any key to continue');
20 1 1:1 133 ($I-*)
21 1 1:1 133 READ(X);
22 1 1:1 141 ($I+*)
23 1 1:1 141 SETCHAIN('GREETING');
24 1 1:1 155 EXIT(PROGRAM);
25 1 1:0 159 END.

```

SYSTEMPOINTSARTUP is present on the Boot disk. It simply tells the analyst when it is time to set up the APM System disk and press a key to continue. When the disk is set up, it transfers control to the GREETING program.

```

1 1 1:D 1 ($$L PRINTER:*)
2 1 1:D 1 ($GREETSHORT TELLS THE ANALYST TO PLACE THE REAL APM SYSTEM DISK IN DRIVE#1*)
3 1 1:D 1 ($RONALD G. SHAPIRO V 2.0 10/19/82*)
4 1 1:D 1
5 1 1:D 1 PROGRAM GREETING;
6 28 1:D 3
7 28 2:D 1 PROCEDURE SETCHAIN(TITLE:STRING);
8 28 3:D 1 PROCEDURE SETCVAL(VAL:STRING);
9 28 4:D 1 PROCEDURE GETCVAL(VAR VAL:STRING);
10 28 5:D 1 PROCEDURE SWAPON;
11 28 6:D 1 PROCEDURE SWAPOFF;
12 28 6:D 1
13 1 1:D 1 USES CHAINSTUFF;
14 1 1:D 3 VAR
15 1 1:D 3 X:CHAR;
16 1 1:D 0 BEGIN
17 1 1:D 0 PAGE(OUTPUT);
18 1 1:D 15 WRITELN('Please insert the APM SYSTEM DISK in drive # 1');
19 1 1:D 81 WRITELN(' Then press any key to continue');
20 1 1:D 133 ($$I-*)
21 1 1:D 133 READ(X);
22 1 1:D 141 ($$I+*)
23 1 1:D 141 SETCHAIN('GREETING');
24 1 1:D 155 EXIT(PROGRAM);
25 1 1:D 159 END.

```

**GREETSHORT**—If the analyst fails to set up the system disk, then the **GREETSHORT** program is executed. It, once again, asks the analyst to set up the system disk and press a key. The only ways to exit from this program are to set up the system disk, press Control Reset or turn the computer off.

```

1 1 1:D 1 ($$L PRINTER: $)
2 1 1:D 1 PROGRAM BLOCKINSTRUCTIONS;
3 1 1:D 3 ($$Program to take text instructions file and convert it to blocked instr file$)
4 1 1:D 3 ($ After editing file, X BLOCKINSTR. At the pause, place this disk in$)
5 1 1:D 3 ($ Drive #1 and place the APM UTIL disk in Drive #2. Press any key.$)
6 1 1:D 3 ($ Within a few minutes, files will be blocked.$)
7 1 1:D 3 ($Note:
8 1 1:D 3 Each frame of text must be exactly 20 lines long in the text file!$)
9 1 1:D 3 ($Ronald G. Shapiro U2.0 10/25/82$)
10 1 1:D 3
11 1 1:D 3 TYPE
12 1 1:D 3 INSTRFILE=RECORD
13 1 1:D 3 LINE:ARRAY[1..20] OF STRING[80];
14 1 1:D 3 END;
15 1 1:D 3
16 1 1:D 3 VAR
17 1 1:D 3 INSTRFILE:FILE OF INSTRFILE;
18 1 1:D 1123 ORIGINST:TEXT;
19 1 1:D 1424 I,J,K,L,M,N:INTEGER;
20 1 1:D 1430 LINE:STRING[80];
21 1 1:D 1471 A1CHAR;
22 1 1:D 1472
23 1 1:0 0 BEGIN
24 1 1:1 0 writeln('press any key to begin');
25 1 1:1 68 read(a);
26 1 1:1 79 REWRITE(INSTRFILE,'apout1:INSTRUCT');
27 1 1:1 106 SEEK(INSTRFILE,1);
28 1 1:1 114 PUT(INSTRFILE);
29 1 1:1 121 CLOSE(INSTRFILE,LOCK);
30 1 1:1 129 RESET(INSTRFILE,'apout1:INSTRUCT');
31 1 1:1 156 RESET(ORIGINST,'apagg4:INSTR.TEXT');
32 1 1:1 186
33 1 1:1 186 J:=1;
34 1 1:1 190 REPEAT
35 1 1:2 190 J:=J+1;
36 1 1:2 198 FOR I:=1 TO 20 DO
37 1 1:3 215 BEGIN
38 1 1:3 215 ($$R-$)
39 1 1:3 215 ($$I-$)
40 1 1:4 215 READLN(ORIGINST,LINE);
41 1 1:4 231 WRITELN(LINE);
42 1 1:4 247 ($$I+$)
43 1 1:4 247 ($$R+$)
44 1 1:4 247 INSTRFILE^.LINE[I]:=LINE;
45 1 1:3 263 END;
46 1 1:2 273 SEEK(INSTRFILE,J);
47 1 1:2 283 PUT(INSTRFILE);
48 1 1:1 290 UNTIL EOF(ORIGINST);
49 1 1:0 300 END.

```

BLOCKINSTR—Blocks the instruction data set for use with the APM package. By using a blocked data set, processing is speeded.

```

1 1 11D 1 (SOL PRINTER: S)
2 1 11D 1 (SProgram to take text help file and convert it to blocked help fileS)
3 1 11D 1 (S After editing file, X BLOCKHELP. At the pause, place this disk inS)
4 1 11D 1 (S Drive #1 and place the APM UTIL disk in Drive #2. Press any key.S)
5 1 11D 1 (S the filenames are: #4:Help1.text, #4:Help2.text, #4:Help3.text or,S)
6 1 11D 1 (S you may use the BRIEFHELP files insteadS)
7 1 11D 1 (S Within a few minutes, files will be blocked.S)
8 1 11D 1 (SNote:
9 1 11D 1 Each frame of text must be exactly 10 lines long in the text file!S)
10 1 11D 1 (SRonald B. Shapiro U2.0 10/25/82S)
11 1 11D 1 PROGRAM BLOCKHELP;
12 1 11D 3
13 1 11D 3 TYPE
14 1 11D 3 HELPFILERECORD
15 1 11D 3 LINE:ARRAY[1..10] OF STRING[80];
16 1 11D 3 END;
17 1 11D 3
18 1 11D 3 VAR
19 1 11D 3 HELPFILERECORD OF HELPFILERECORD;
20 1 11D 713 ORIGHELP:TEXT;
21 1 11D 1014 I,J,K,L,M,N:INTEGER;
22 1 11D 1020 LINE:STRING[80];
23 1 11D 1041 FILENAME: STRING[80];
24 1 11D 1102 A:CHAR;
25 1 11D 1103
26 1 11D 0 BEGIN
27 1 11D 0 WRITELN('Pause--set up disks--then anykey (return)');
28 1 11D 87 READLN;
29 1 11D 95 J:=0;
30 1 11D 99 REWRITE(HELPFILERECORD,'#5:HELP');
31 1 11D 118 CLOSE(HELPFILERECORD,PURGE);
32 1 11D 126 REWRITE(HELPFILERECORD,'#5:HELP');
33 1 11D 145 SEEK(HELPFILERECORD,1);
34 1 11D 153 PUT(HELPFILERECORD);
35 1 11D 160 CLOSE(HELPFILERECORD,LOCK);
36 1 11D 168 RESET(HELPFILERECORD,'#5:HELP');
37 1 11D 187 REPEAT
38 1 11D 187 WRITE('Input Filename (esc if done): ');
39 1 11D 229 readln(filename);
40 1 11D 249 IF (ORD(FILENAME[1])=27) THEN

```

BLOCKHELP--Blocks the HELP (and BRIEFHELP) data set for use with the APM package. By using a blocked data set, processing is speeded.

```

41 1 1:3 259 EXIT(PROGRAM);
42 1 1:2 263 RESET(ORIGHELP,filename);
43 1 1:2 276
44 1 1:2 276 REPEAT
45 1 1:3 276 J:=J+1;
46 1 1:3 284 FOR I:=1 TO 10 DO
47 1 1:4 301 BEGIN
48 1 1:4 301 ($R-8)
49 1 1:4 301 ($I-8)
50 1 1:5 301 READLN(ORIGHELP,LINE);
51 1 1:5 317 M:=0;
52 1 1:5 321 FOR K:=80 DOWNT0 1 DO
53 1 1:6 338 BEGIN
54 1 1:7 338 A:=LINE[K];
55 1 1:7 348 IF (ORD(A)<29)OR(ORD(A)>127) THEN
56 1 1:8 361 BEGIN
57 1 1:9 361 DELETE(LINE,K,1);
58 1 1:8 371 END;
59 1 1:6 371 END;
60 1 1:5 381 WRITELN(LINE);
61 1 1:5 397 ($I+8)
62 1 1:5 397 ($R+8)
63 1 1:5 397 IF J>0 THEN HELPFILE^.LINE[I]:=LINE;
64 1 1:4 420 END;
65 1 1:3 430 IF J>0 THEN
66 1 1:4 437 SEEK(HELPFILE,J);
67 1 1:3 447 IF J>0 THEN
68 1 1:4 454 PUT(HELPFILE);
69 1 1:2 461 UNTIL EOF(ORIGHELP);
70 1 1:2 471 close(orighelp);
71 1 1:1 480 until (ord(filename[1])=27);
72 1 1:0 490 END.

```

See previous page for program description.

```

1 1 11D 1 (SSL PRINTER:8)
2 1 11D 1 (RUNNING THE FOLLOWING PROGRAM MODIFIES THE SYSTEM.APPLE FILE FOR USE W/VIDEX8)
3 1 11D 1 PROGRAM VIDPATCH;
4 1 11D 3 VAR BUF:PACKED ARRAY[0..31,0..511] OF 0..255;
5 1 11D 8195 F:FILE;
6 1 11D 8235 I:INTEGER;
7 1 11D 0 BEGIN
8 1 111 0 RESET(F,'04:SYSTEM.APPLE');
9 1 111 43 I:=BLOCKREAD(F,BUF,32);
10 1 111 65 CLOSE(F);
11 1 111 74
12 1 111 74 BUF[3,389]:=160;
13 1 111 100 BUF[3,390]:=48;
14 1 111 124 BUF[3,394]:=40;
15 1 111 148 BUF[3,455]:=173;
16 1 111 174 BUF[3,456]:=0;
17 1 111 198 BUF[3,457]:=192;
18 1 111 224 BUF[3,458]:=16;
19 1 111 248 BUF[3,459]:=29;
20 1 111 272 BUF[3,460]:=32;
21 1 111 296 BUF[3,461]:=24;
22 1 111 320 BUF[3,462]:=218;
23 1 111 346 BUF[3,463]:=234;
24 1 111 372 BUF[4,207]:=3;
25 1 111 396 RESET(F,'04:SYSTEM.APPLE');
26 1 111 424 I:=BLOCKWRITE(F,BUF,32);
27 1 111 446 CLOSE(F);
28 1 11D 455 END.

```

VIDPATCH (written by VIDEX) updates the SYSTEM.APPLE program for use with the VIDEX board. This program must be run once with each SYSTEM.APPLE file.